Konzeption einer Verkehrsnetzrepräsentation für kognitive Agenten in virtuellen Umgebungen

Tobias Haubrich

Publisher: Dean Prof. Dr. Wolfgang Heiden

University of Applied Sciences Bonn-Rhein-Sieg,
Department of Computer Science

Sankt Augustin, Germany

July 2013

Technical Report 01-2013

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

ISSN 1869-5272

Zusammenfassung

Als Basis für Simulationen innerhalb virtueller Umgebungen werden meist unterliegende Semantiken benötigt. Im Fall von Verkehrssimulationen werden in der Regel definierte Verkehrsnetzwerke genutzt. Die Erstellung dieser Netzwerke wird meist per Hand durchgeführt, wodurch sie fehleranfällig ist und viel Zeit erfordert. Dieses Projekt wurde im Rahmen des AVeSi Projektes durchgeführt, in dem an der Entwicklung einer realistischen Verkehrssimulation für virtuelle Umgebung geforscht wird. Der im Projekt angestrebte Simulationsansatz basiert auf der Nutzung von zwei Komplexitätsebenen – einer mikroskopischen und einer mesoskopischen. Um einen Übergang zwischen den Simulationsebenen zu realisieren ist eine Verknüpfung der Verkehrsnetzwerke notwendig, was ebenfalls mit einem hohen Zeitaufwand verbunden ist. In diesem Bericht werden Modelle für Verkehrsnetzwerke beider Ebenen vorgestellt. Anschließend wird ein Ansatz beschrieben, der eine automatische Generierung und Verknüpfung von Verkehrsnetzwerken beider Modelle ermöglicht. Als Grundlage für die Generierung der Netzwerke dienen Daten im OpenDRIVE®-Format. Zur Evaluierung wurden wirklichkeitsgetreue OpenStreetMap-Daten, durch Verwendung einer Drittanbietersoftware, in OpenDRIVE®-Datensätze überführt. Es konnte nachgewiesen werden, dass es durch den Ansatz möglich ist, innerhalb weniger Minuten, große Verkehrsnetzwerke zu erzeugen, auf denen unmittelbar Simulationen ausgeführt werden können. Die Qualität der zur Evaluierung generierten Netzwerke reicht jedoch für Umgebungen, in denen ein hoher Realitätsgrad gefordert wird, nicht aus, was einen zusätzlichen Bearbeitungsschritt notwendig macht. Die Qualitätsprobleme konnten darauf zurückgeführt werden, dass der Detailgrad, der den Evaluierungsdaten zu Grunde liegenden OpenStreetMap-Daten, nicht hoch genug und der Überführungsprozess nicht ausreichend transparent ist.

Inhaltsverzeichnis

Abbildungsverzeichnis vi

Tabellenverzeichnis vii

1. Einleitung 1
- 1.1. Problemstellung / Motivation . . . 1
- 1.2. Fragestellung / Ziele . . . 1
- 1.3. Semantic Road Network Models for Rapid 3D Traffic Scenario Generation . . . 2

2. Grundlagen für virtuelle Verkehrsnetzwerke und -simulation 3
- 2.1. Ansätze zur Verkehrssimulation . . . 3
- 2.2. OpenDRIVE® . . . 5
  - 2.2.1. Grundlegende Einträge (Records) . . . 5
  - 2.2.2. Aufbau von Straßen in OpenDRIVE® . . . 7
  - 2.2.3. Verknüpfung von Straßen in OpenDRIVE® . . . 7
- 2.3. Straßenbeschreibung / Trassierung . . . 8
  - 2.3.1. Gerade . . . 9
  - 2.3.2. Kreisabschnitt . . . 9
  - 2.3.3. Klothoide (engl. Euler Spiral) . . . 9
- 2.4. Verkehrsnetzwerkmodelle für virtuelle Umgebungen . . . 11
  - 2.4.1. Verkehrsnetzwerkmodelle in Videospielen . . . 11
  - 2.4.2. Verkehrsnetzwerkmodelle in Forschungsprojekten . . . 12
- 2.5. Simulationsansatz im AVeSi-Projekt . . . 13

3. Semantische Verkehrsnetzwerkmodelle 15
- 3.1. Mesoskopische Simulationsebene . . . 15
- 3.2. Mikroskopische Simulationsebene . . . 18
  - 3.2.1. Verkehrsnetzwerkmodell für die mikroskopische Simulationsebene . . . 18
  - 3.2.2. Informationen und OpenDRIVE® . . . 21
  - 3.2.3. Zusätzliche Informationen für Junction- und Waypoint-Elemente . . . 27

4. SeRoNet: Generierung semantischer Verkehrsnetzwerke 29
- 4.1. Tools . . . 29
  - 4.1.1. Unity Game Engine . . . 29
  - 4.1.2. Trian3D Builder . . . 30
  - 4.1.3. xodrReader und xodrViewer . . . 30
- 4.2. Objektorientierte Darstellung der OpenDRIVE®-Daten . . . 31
- 4.3. Verkehrsnetzwerkmodell für warteschlangenbasierte Mesosimulation . . . 31
  - 4.3.1. Elemente des Modells für die mesoskopische Simulationsebene . . . 31
  - 4.3.2. Generierungsprozess . . . 32
- 4.4. Verkehrsnetzwerkmodell für kognitive Mikrosimulation . . . 33
  - 4.4.1. Elemente des Modells für die mikroskopische Simulationsebene . . . 33
  - 4.4.2. Generierungsprozess . . . 35

5. Evaluation 40
- 5.1. Evaluierungsstrategie . . . 40
- 5.2. Festgehaltene Leistungsmerkmale . . . 41
- 5.3. Überprüfung der geometrischen Berechnungen . . . 42
- 5.4. Evaluation durch Navigation von Agenten . . . 43
  - 5.4.1. Mesoskopische Simulationsebene . . . 43
  - 5.4.2. Mikroskopische Simulationsebene . . . 45

5.5. Evaluation durch Beobachtung 47
5.5.1. Evaluierungsstrategie und Kriterien 50
5.5.2. Durch Beobachtung gewonnene Informationen 53
5.5.3. Auswertung gewonnener Informationen 59

6. Zusammenfassung **65**
6.1. Ausblick 66
6.1.1. Weiterentwicklung des Rapid Scenario Generation Workflow 66
6.1.2. Zusammenführung der Verkehrsnetzwerkmodelle beider Simulationsebenen 66
6.1.3. Weiterentwicklung der kognitiven Agenten 67

Literaturverzeichnis **68**

A. Anhang **70**

Abbildungsverzeichnis

1. Virtuelle Umgebung innerhalb des FIVIS Simulators. 1
2. Empfohlene Drei-Schichten-Architektur [23] 2
3. Architekturkonzept 2
4. Aufbau einer Straße in OpenDRIVE® [7] 8
5. Verbindung von Straßen durch einen Junction-Eintrag [7] 8
6. Krümmungsverhalten von Strecke, Klothoide und Kreisabschnitt nach [7] 10
7. Beispielhafte Form einer Klothoide 11
8. Konzept eines Connector-Elements 20
9. Konzept eines Lane-Elements 21
10. Konzept eines Road- bzw. Path-Elements 21
11. Konzept eines Junction-Elements 21
12. Visualisierung eines Lane-Elements 34
13. Visualisierung eines Connector-Elements 34
14. Visualisierung eines Waypoint-Elements 34
15. Visualisierung eines Road-Elements 35
16. Visualisierung eines Junction-Elements 35
17. Pseudocode zum Generierungsprozess für Verkehrsnetzwerke der Mikrosimulationsebene 36
18. Vereinigung von Lane-Einträgen nach [7] 37
19. Vergleich von Folgen von Waypoint-Elementen mit und ohne Douglas-Pecker-Reduktion 38
20. Informationen zu den verwendeten OpenDRIVE®-Datensätzen 40
21. Dauer der Generierung von Verkehrsnetzwerken 41
22. Überlagerung von Referenzlinien und zugehörigem 3D-Modell einer Stadt 42
23. Generierte Verkehrsnetzwerke aus Beispieldaten 43
24. Generierte Verkehrsnetzwerke aus erzeugten Daten 43
25. Häufigkeit an Wartevorgängen von Verkehrsteilnehmern 45
26. Verteilung der Anzahl an Abbiegevorgängen pro Verkehrsteilnehmer 46
27. Anzahl der Befahrungen einzelner Road- und Lane-Elemente 48
28. Häufigkeitsverteilung von Verweilzeiten der Agenten im mikroskopischen Netzwerk 49
29. Zur Evaluation gewähltes Szenario 52
30. GoogleMaps Kartenausschnitt zum Evaluierungsszenario 52
31. Generiertes Mikro-Netzwerk zum Evaluierungsszenario 54
32. Generiertes Meso-Netzwerk zum Evaluierungsszenario 54
33. OpenStreetMap Kartenausschnitt überlagert mit dem generierten Verkehrsnetzwerk 55
34. GoogleMaps Kartenausschnitt überlagert mit dem generierten Verkehrsnetzwerk 55
35. Untersuchte Bereiche innerhalb des Szenarios 56
36. Szenario Bereich 1 (Kreuzung) 57
37. Szenario Bereich 2 (Kreuzung) 57
38. Szenario Bereich 3 (Kreisverkehr) 58
39. Szenario Bereich 4 (Kreisverkehr) 58
40. Szenario Bereich 5 (Autobahnabschnitt) 60
41. Arbeitsablauf zur Realisierung einer schnellen Szenario Generierung [12] 67

Tabellenverzeichnis

1. Mögliche Lane Description Records [7] 7
2. Parameter zur eindeutigen Bestimmung einer Geraden [14] 9
3. Parameter zur eindeutigen Bestimmung eines Kreisabschnittes [14] 9
4. Parameter zur eindeutigen Bestimmung einer Klothoiden [14] 10
5. Informationen für Kanten 16
6. Informationen für Knoten 17
7. Informationen für Lane-Elemente 22
8. Informationen für Connector-Elemente 24
9. Informationen für Road- bzw. Path-Elemente 25
10. Informationen für Junction-Elemente 26
11. Informationen für Waypoint-Elemente 26
12. Durch MonoBehaviour bereitgestellte Funktionen 30
13. Übersicht der verwendeten OpenDRIVE®-Datensätze 40
14. Eigenschaften des für die Evaluation verwendeten Computers 41
15. Auf Basis eines 10 Stunden langen Testlaufs ermittelte Eigenschaften zur mesoskopischen Simulationsebene 44

1. Einleitung

1.1. Problemstellung / Motivation

Im Rahmen des FIVIS[1] Forschungsprojektes am Institut für Visual Computing der Hochschule Bonn-Rhein-Sieg wird ein Fahrradfahrsimulator entwickelt. Ein wesentliches Ziel ist es, den Simulator für einen gezielten Einsatz in der Verkehrserziehung und im Verkehrssicherheitstraining weiterzuentwickeln. In Abbildung 1 ist die virtuelle Umgebung innerhalb des FIVIS Simulators dargestellt.

Abbildung 1: Virtuelle Umgebung innerhalb des FIVIS Simulators.

Im AVeSi[2] Projekt soll nun eine realistische Verkehrssimulation für einen Einsatz in dem FIVIS System entwickelt werden. Dabei sollen auf Basis von Agenten mit kognitiven Eigenschaften auch Gefahrensituationen entstehen können, welche bei bestehenden Verkehrssimulationen größtenteils nicht berücksichtigt werden. Im Simulator wird dazu ein Netzwerk von virtuellen Straßen aufgebaut, auf dem sich die Agenten sowie der Fahrer fortbewegen. Zu einem existierenden virtuellen Straßennetz liegen Straßenbeschreibungen im OpenDRIVE®-Dateiformat vor. Die im XML-Format hinterlegten Daten sind in dieser Form zunächst nicht direkt nutzbar. Die Aufgabe ist nun die Informationen der Dateien so aufzubereiten, dass sie in die Simulationsumgebung integriert und dadurch von den Verkehrsagenten genutzt werden können. Die XML-basierten Dateien müssen dazu eingelesen und anschließend ausgewertet werden. Um dies zu ermöglichen, muss eine Schnittstelle zum Zugriff auf diese Daten erstellt und anschließend ein Konzept entwickelt werden, wie die Agenten diese Daten möglichst effektiv nutzen können. In Abbildung 2 ist die von den OpenDRIVE®Entwicklern vorgeschlagene Verwendung des Formates abgebildet. An dieser Stelle wird eine Drei-Schichten-Architektur empfohlen. Auf Schicht 1 befindet sich die Straßenbeschreibung, welche in einer statischen OpenDRIVE®-Datei hinterlegt ist. Zum Zugriff darauf wird in der zweiten Schicht eine Auswertungsbibliothek definiert. Auf die Auswertungsbibliothek kann nun, von den einzelnen Bereichen der obersten Schicht, unabhängig zugegriffen werden.

1.2. Fragestellung / Ziele

In diesem Kontext ergeben sich einige, folgend erläuterte, noch zu lösende Fragestellungen: Wie lässt sich die gegebene Repräsentation der Straßendaten in FIVIS integrieren? Dazu muss geklärt werden, welche der zahlreichen in der OpenDRIVE®-Spezifikation beschriebenen Informationen für die Berechnung der Simulation relevant sind. Zusätzlich muss eine passende Darstellung für diese Daten gefunden und eine Schnittstelle zum Zugriff darauf definiert werden. Die Repräsentation der Daten soll als Grundlage für die Entwicklung kognitiver Verkehrsagenten dienen. Dabei ist die Frage zu klären, ob an dieser Stelle eine Wissensrepräsentation der Agenten in die Verkehrsnetzrepräsentation integriert werden soll oder ob es sinnvoller ist, diese an einer anderen Stelle zu hinterlegen.

[1] Entwicklung eines Fahrradfahrsimulators zur Verkehrserziehung und zum Verkehrssicherheitstraining für verschiedene Altersklassen.

[2] Agentenbasierte Verkehrssimulation mit psychologischen Persönlichkeitsprofilen.

Ziel dieser Arbeit ist es nun erstens, eine Schnittstelle zum Zugriff auf die zum virtuellen Verkehrsnetzwerk gehörenden und im OpenDRIVE®-Format hinterlegten Daten zu definieren und zweitens, darauf aufbauend ein Konzept zu entwickeln, wie die Daten möglichst effizient von den Agenten genutzt werden können.

1.3. Semantic Road Network Models for Rapid 3D Traffic Scenario Generation

Ein Teil dieser Arbeit wurde bereits im Rahmen des Workshops der ASIM/GI-Fachgruppen STS und GMMS im Jahr 2013 publiziert. Die dazugehörige Veröffentlichung trägt den Titel: *Semantic Road Network Models for Rapid 3D Traffic Scenario Generation* [12].

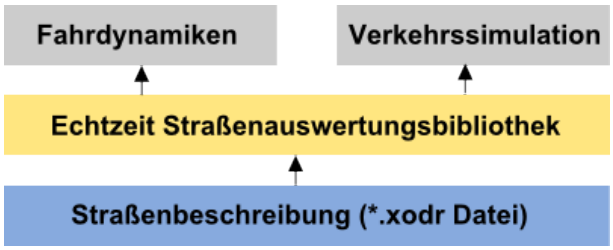


Abbildung 2: Empfohlene Drei-Schichten-Architektur zur Nutzung des OpenDRIVE®-Formats. Abbildung nach [23].

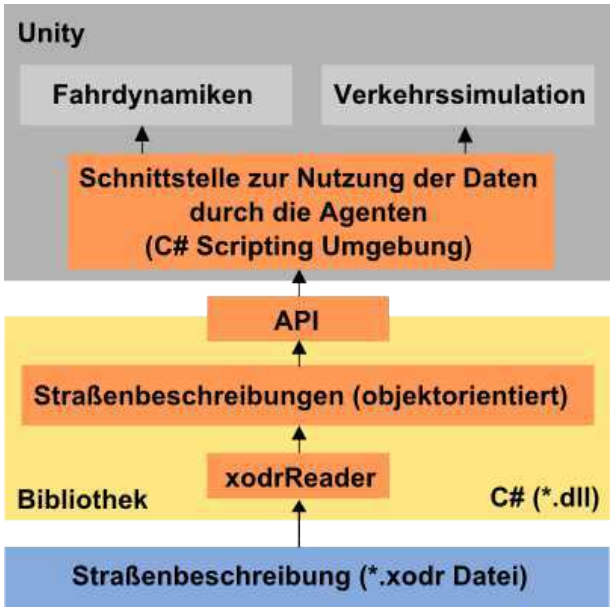


Abbildung 3: Architekturkonzept aufbauend auf der in Abbildung 2 empfohlenen Architektur.

2. Grundlagen für virtuelle Verkehrsnetzwerke und -simulation

In diesem Kapitel wird auf einige für diese Arbeit relevante Grundlagen eingegangen. Zu Beginn des Kapitels werden drei etablierte Verkehrssimulationsansätze vorgestellt. Anschließend werden einige Aspekte diskutiert, die sich mit der Beschreibung und dem Aufbau von Verkehrsnetzwerken in virtuellen Umgebungen befassen. Begonnen wird mit der OpenDRIVE®-Spezifikation welche ein offenes Format für die digitale Beschreibung von Verkehrsnetzwerken definiert. Dafür bietet sie Beschreibungsmöglichkeiten für alle Hauptmerkmale echter Straßennetzwerke an. Die Spezifikation soll als Grundlage für die Generierung von Verkehrsnetzen für das AVeSi-Projekt fungieren. Danach wird auf ein paar mathematische Grundlagen für Straßenbeschreibung bzw. Trassierung eingegangen. Die eingeführten Methoden werden sowohl bei der Straßenplanung und dem Straßenbau als auch zur Beschreibung von Straßen im OpenDRIVE®-Format verwendet. Anschließend werden einige bestehende und in der Literatur beschriebene Verkehrsnetzwerkmodelle betrachtet. Dazu werden unter anderem deren Eigenheiten und Anwendungsgebiete betrachtet. Letztendlich wird noch auf den momentan innerhalb des AVeSi-Projekts angestrebten Simulationsansatz eingegangen.

2.1. Ansätze zur Verkehrssimulation

In der klassischen Verkehrssimulation werden drei grundlegende Simulationsansätze unterschieden. Zum einen gibt es den makroskopischen Simulationsansatz welcher einen Top-Down-Ansatz darstellt. Der Verkehr wird von oben herab betrachtet und es wird versucht auf Basis von mathematischen Formeln realistische Verkehrsflüsse zu erzeugen ohne dabei detaillierte Elemente des Straßenverkehrs modellieren zu müssen. Dagegen steht der mikroskopische Ansatz, welcher ein Bottom-Up-Ansatz ist. Hier werden einzelne Verkehrsteilnehmer während der Simulation einer individuellen Betrachtung unterzogen. Jeder dieser Verkehrsteilnehmer besitzt eine Menge an Eigenschaften, die Aspekte wie Aussehen und Verhalten betreffen. Bei mikroskopischen Simulationen sollen letztendlich durch das Zusammenspiel vieler eigenständig simulierter Verkehrsteilnehmer realistische Verkehrsflüsse entstehen. [9]

Zusätzlich existieren hybride Ansätze wie z.B. mesoskopische Verkehrssimulation. In diesen wird versucht Vorteile der beiden anderen Ansätze zu vereinen. Die drei eingeführten Simulationsansätze werden in den folgenden Abschnitten genauer betrachtet. Anschließend wird ein kurzer Vergleich zwischen deren Eigenschaften gezogen.

Makrosimulation

Im makroskopischen Simulationsansatz werden Verkehrsteilnehmer nicht individuell sondern kollektiv betrachtet und berechnet. Es wird versucht durch einen Top-Down-Ansatz auf Basis von mathematischen Formeln, welche z.T. von Formeln zur Simulation von Flüssigkeiten oder Ähnlichem abgeleitet sind, eine realistische Simulation zu erzeugen. In diesem Ansatz wird mit ausdrucksstarken Kennwerten wie z.B. Verkehrsdichte und Verkehrsfluss gearbeitet. Durch ein solches Vorgehen wird es vermieden schwierig zu modellierende Prozesse wie z.B. Spurwechsel, Überholvorgänge oder Vorfahrtsregelungen aufwändig nachbilden zu müssen. Stattdessen werden diese auf einer sehr abstrakten Ebene betrachtet und können höchstens durch Anpassung von Formelparametern beeinflusst werden. Ein Vorteil eines solchen Simulationsansatzes ist es, dass der Rechenaufwand stark eingeschränkt wird was wiederum dazu führt, dass Makrosimulationen besonders bei Echtzeitanwendungen bevorzugt werden. Sie können z.B. verwendet werden um Verkehrsnetze auf Engstellen hin zu untersuchen oder Stauvorhersagen zu treffen. Durch das Fehlen der Individualität eigenen sich solche Ansätze jedoch nicht für die Nachbildung von realitätsnahen Situationen welche z.B. in virtuellen Umgebungen oder Videospielen angestrebt werden. Beispiele für makroskopische Simulationsansätze sind LWR [17][20], GTK [13] oder Aw-Rascle [2].

Mikrosimulation

Im Gegensatz zur Makrosimulation wird in der Mikrosimulation jeder Verkehrsteilnehmer als eigenständige Entität mit individuellen Eigenschaften betrachtet. Es wird versucht von unten herauf eine realistische Simulation zu erzeugen in dem einzelne Verkehrsteilnehmer modelliert werden. Der Realismus soll hier letztendlich durch das Zusammenspiel der einzelnen Komponenten entstehen. Bei einer Mikrosimulation können verschiedenste Sachverhalte auf einer sehr niedrigen Stufe detailgetreu betrachtet und modelliert werden. Dies können z.B. Abbiegevorgänge, Spurwechsel oder Vorfahrtsregelungen sein. Bei diesem Ansatz ist es auch möglich die Interaktionen von Verkehrsteilnehmern im Bezug zu anderen Verkehrsteilnehmern zu berücksichtigen. Unter der Voraussetzung, dass ein entsprechender Aufwand in das Design der Simulation investiert wird, ist es durch diesen Ansatz möglich Simulationen mit beliebigem Realitätsgrad zu erzeugen. Der große Nachteil ist jedoch, dass der Berechnungsaufwand um ein vielfaches steigt je mehr und je detaillierter einzelne Aspekte berücksichtigt werden bzw. je mehr Verkehrsteilnehmer simuliert werden. Dies hat zur Folge, dass auf Grund von begrenzter Rechenleistung und erst recht bei Echtzeitanwendungen nur kleine Gebiete durch einen mikroskopischen Ansatz betrachtet werden können. Ein Beispiel für ein mikroskopisches Verkehrsflussmodell stellt das Intelligent-Driver Model (IDM) [21] dar.

Mesosimulation / queue micro-simulation

Der Begriff der Mesosimulation, teilweise auch queue micro-simulation [19] genannt, ist im Gegensatz zu den anderen beiden Ansätzen nur schwach definiert. Im allgemeinen Fall stellt er eine Kombination des Mikroskopischen- und Makroskopischen Ansatz, also einen hybriden Ansatz dar. Ein Ziel ist es z.B. den Rechenaufwand einzuschränken jedoch weiterhin eigenständige Entitäten betrachten zu können. Ansätze für Mesosimulationen sind Warteschlangenmodelle (queueing models) [8], zelluläre Automaten [18] oder Partikelsysteme. Von Seiten der Komplexität und des Realismus aus betrachtet sind Mesosimulationsansätze zwischen den beiden anderen Ansätzen einzuordnen. Es existieren z.B. Ansätze die Realismus nahe einer Mikrosimulation erzeugen und gleichzeitig den Rechenaufwand stark einschränken [8]. Ein Beispielprojekt ist der Microscopic Multi-Agent Traffic Simulator (MMTS)[3] mit dem u.A. bereits das Verkehrsnetz der Schweiz simuliert wurde. Dabei wurden jedoch nur viel befahrene Straßen wie Autobahnen und Bundesstraßen betrachtet. In [19] wird erwähnt, dass diese Simulation vom Aufwand her ungefähr mit der einer kompletten Stadt wie London oder Los Angeles vergleichbar ist. Ein Beispiel für ein Framework, dass einen Mesosimulationsanatz verwendet ist MATSIM[4].

Vergleich

Die Wahl des Simulationsansatzes wirkt sich auf Faktoren wie Komplexität, Kosten, Präzision und Realismus der Simulation aus. So können durch mikroskopische Simulation sehr viele Aspekte auf einer detaillierten Stufe simuliert werden um einen hohen Realismus zu erzeugen. Dies erfordert jedoch sehr viel Rechenaufwand. Mit makroskopischen Ansätzen ist es möglich z.T. sehr große Gebiete auf einer abstrakten Ebene zu betrachten und dadurch die Simulation relativ kostengünstig zu gestalten. Solche Ansätze werden z.B. genutzt um Stauvorhersagen treffen zu können oder präventiv geplante Straßennetze auf Schwachstellen zu untersuchen. Mesoskopische Ansätze versuchen die Individualität von Verkehrsteilnehmern zu bewahren jedoch gleichzeitig die Abstraktionsstufe so zu wählen, dass es ermöglicht wird größere Gebiete zu betrachten als es bei Mikrosimulationen möglich ist.

[3] Von K. Nagel im Rahmen eines Projektes an der ETH Zurich entwickelt.
[4] `http://www.matsim.org`

2.2. OpenDRIVE®

Die OpenDRIVE®-Spezifikation stellt einen Standard zur Beschreibung von spurbasierten Verkehrsnetzwerken zur Verfügung. Über den Standard können die Geometrie von Straßen sowie die Eigenschaften von Elementen an den Straßen beschrieben werden [7]. In den folgenden Unterabschnitten werden die wichtigsten Elemente der Spezifikation eingeführt und deren Aufbau und Zusammenhang dargestellt. Genauere Informationen können der OpenDRIVE®-Formatspezifikation ([7]) entnommen werden. In dieser Arbeit wird die Version 1.3D der Spezifikation verwendet.

2.2.1. Grundlegende Einträge (Records)

Die Beschreibung von Straßen geschieht durch verschiedene Formen von Einträgen (Records) innerhalb einer XML-basierten *.xodr-Datei. An dieser Stelle werden die grundlegenden in der Spezifikation definierten Eintragsformen erläutert. Im folgenden Kapitel wird dann der Zusammenhang zwischen diesen näher erklärt. Die von der Hierarchie am höchsten liegenden Einträge sind: Header, Road, Controller und Junction. Ein Header-Eintrag beinhaltet allgemeine Informationen zu der Datei, besitzt jedoch keine Kindeinträge und wird deshalb folgend nicht näher betrachtet.

Road-Eintrag

Straßen werden grundsätzlich durch Road-Einträge und deren Kindeinträge beschrieben. Ein Road-Eintrag enthält Parameter wie Name, Länge und ID. Außerdem kann in Straßen, welche zu einer Kreuzung (Junction) gehören die ID des zugehörigen Junction-Eintags gespeichert werden. Straßen die solch einen Verweis enthalten (also zu einer Kreuzung gehören) werden Path bzw. Pfad genannt. Weiterhin besitzt jeder Road-Eintrag eine Menge von Kindeinträgen von denen einige folgend beschrieben sind.

- *Link-Eintrag*
  Verbindungen werden über Link-Einträge angegeben. Diese besitzen Kindeinträge mit IDs, über welche vorangehende und nachfolgende Road-Einträge referenziert werden.

- *Type-Eintrag*
  Über einen Type-Eintrag kann die Kategorie der Straße angegeben werden. Dieser Eintrag besitzt keine weiteren Kindeinträge.

- *Planview- / Geometry-Eintrag (Ebene)*
  Zur Beschreibung der Geometrie einer Straße wird der Planview-Eintrag und sein Kindeintrag Geometry verwendet. Zu beachten ist, dass an dieser Stelle nur die Geometrie der Straße in der Ebene beschrieben wird. Heißt, dass an dieser Stelle erst einmal keine Steigung definiert wird. Immer wenn sich die Geometrieform ändert, muss ein neuer Geometry-Eintrag angegeben werden, in welchem dessen Beginn über eine s-Koordinate festgelegt wird. Zusätzlich werden in Geometry-Einträgen die Koordinaten des Startpunktes, sowie die Ausrichtung und die Länge des Abschnittes angegeben. Um die restlichen benötigten Informationen zur Verfügung zu stellen, wird je nach Geometrieform des Straßenabschnittes einer der drei möglichen Kindeinträge (Line, Spiral oder Arc) verwendet. Auf die geometrische Beschreibung von Straßen wird in Kapitel 2.3 noch ausführlicher eingegangen.

- *Planview- / ElevationProfile-Eintrag (Steigung)*
  Zur Beschreibung der Steigung werden ElevationProfile-Einträge verwendet. Die Steigung wird dabei über ein Polynom dritten Grades beschrieben.

- *Planview- / LateralProfile-Eintrag (Neigung)*
  Zur Beschreibung der Neigung werden LateralProfile-Einträge verwendet. Es können zwei

verschiedene Arten von Neigungen beschrieben werden. Zum einen Superelevation, also die komplette Straße fällt zu einer Seite hin ab und zum anderen Crossfall. Bei letzterem fällt die Straße von der Referenzlinie aus zu beiden Seiten ab.

- *Lanes-Eintrag*
  Spuren werden im Lanes-Eintrag definiert. Wie dies genau geschieht wird im Abschnitt 2.2.2 genauer erläutert.

- *Objects-Eintrag*
  Über den Road Objects Record können neben der Straße (oder auch über der Straße) befindliche Objekte unterschiedlicher Formen beschrieben werden.

- *Signals-Eintrag*
  Im Road Signals Record können Signale die einen Bezug zur Straße haben beschrieben werden. Angegeben werden können Informationen wie Name, Orientierung und die Position im Verhältnis zur Straße.

- *Surface-Eintrag*
  Über den Surface-Eintrag können CRG (Curved Regular Grid) Daten eingebunden werden. OpenCRG ist ein eigener Standard zur Beschreibung von Straßenoberflächen.

In den beiden anschließenden Abschnitten (2.2.2 und 2.2.3), wird auf zwei wichtige Aspekte bei der Straßenbeschreibung noch einmal genauer eingegangen. Erstens den Aufbau und zweitens die Verknüpfung von Straßen.

Controller-Eintrag

Über Controller-Einträge können Gruppen von Signalen koordiniert werden. Jeder Controller-Eintrag enthält eine Anzahl von Control-Einträgen welche einen Verweis auf einen Signal-Eintrag besitzen.

Junction-Eintrag

Die Beschreibung von Kreuzungen erfolgt im Junction Record. Hier können Listen mit möglichen Verbindungen innerhalb einer Kreuzung definiert werden. Zusätzlich können Straßen priorisiert und zur Kreuzung gehörige Controller-Einträge hinterlegt werden. Der Aufbau von Straßen ist in Kapitel 2.2.3 noch einmal genauer erläutert.

Zusätzliche Eintragsformen

Es existieren noch drei zusätzliche Eintragsformen, welche an jeder Stelle einer OpenDRIVE®-Datei eingefügt sein können.

- *UserData-Eintrag*
  Über UserData-Einträge können bei Bedarf zusätzlich eigens definierte Informationen in OpenDRIVE®-Dateien eingefügt werden. So können eigene Funktionalitäten oder fehlende Informationen ergänzt werden.

- *Include-Eintrag*
  Über Include-Einträge können wie gewohnt Dateien in andere Dateien inkludiert werden. Dies kann z.B. zur Übersichtlichkeit beitragen.

Record	Beschreibung
Lane Width Record	Breite der Straße
Road Mark Record	Äußere Straßenmarkierung
Lane Material Record	Straßenuntergrund bzw. Material
Lane Visibility Record	Sichtbarkeit des Umfeldes
Lane Speed Record	Geschwindigkeitsbegrenzung
Lane Access Record	Zugangsbeschränkungen für einzelne Fahrzeugtypen
Lane Height Record	Höhe des einzelnen Spurabschnitts

Tabelle 1: Mögliche Lane Description Records [7].

- *Set-Eintrag*
  Über Set-Einträge können für bestimmte Bereiche alternative Informationen beschrieben werden. Für jeden Informationssatz muss ein Instance-Kindeintrag existieren. Anwendungen wären hier z.B. einen Straßenabschnitt einmal mit und einmal ohne Baustelle zu beschreiben.

2.2.2. Aufbau von Straßen in OpenDRIVE®

In OpenDRIVE® werden Straßen durch unterschiedliche Einträge (Records) innerhalb einer XML-basierten *.xodr-Datei beschrieben. Auf zwei für das Projekt besonders wichtige Eintragsformen soll folgend genauer eingegangen werden:

- Road Geometry Header Record
- Road Lane Section Record

Über den *Geometry Record*, der Teil des *Road Plan View Records* ist, können unterschiedliche Geometrieabschnitte auf der Straße definiert werden. Durch Aneinanderreihung von drei verschiedenen geometrischen Formen (Line, Arc und Spiral), welche in Kapitel 2.3 näher beschrieben werden, wird eine Referenzlinie erzeugt welche den Verlauf der Straße widerspiegelt. [7]

Parallel dazu wird die Straße durch LaneSection-Einträge untergliedert. In jedem LaneSection-Eintrag ist die Anzahl der Spuren (Lane-Einträge) konstant. Die Eigenschaften jeder Spur können sich jedoch ändern. Jede Spur besitzt eine innerhalb ihres LaneSection-Eintrags eindeutige Nummer. Die Referenzlinie besitzt die Nummer 0 und darf laut Spezifikation keine Breite besitzen. Links der Referenzlinie werden die Spuren aufsteigend definiert also 1,2,... und Rechts abfallend -1,-2,.... Abbildung 4 veranschaulicht den Aufbau von Straßen noch einmal. [7]

Zur Beschreibung weiterer Eigenschaften kann zu jeder Spur (innerhalb eines LaneSection-Eintrags) eine Anzahl verschiedener anderer Einträge erstellt werden. In Tabelle 1 ist eine Liste der verschiedenen Eintragsarten zu sehen. Die dort aufgeführten Records werden folgend zusammengefasst als *Lane Description Records* bezeichnet.

2.2.3. Verknüpfung von Straßen in OpenDRIVE®

Die Verknüpfung von Straßen kann auf zwei Weisen erfolgen. Erstens können Straßen direkt verbunden werden. Hierbei wird innerhalb eines Road-Eintrags der Eintrag zur Vorgänger- bzw. Nachfolgerstraße direkt über einen Predecessor- bzw. Successor-Eintrag referenziert. Es wird zusätzlich angegeben ob der Start- oder Endpunkt der anderen Straße angrenzt. Die Verbindung der Spuren wird über einen Link-Eintrag innerhalb eines LaneSection-Eintrags angegeben.

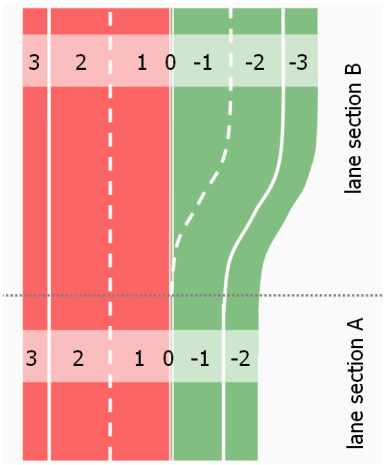


Abbildung 4: Aufbau einer Straße in OpenDRIVE®. Bildquelle: [7, S.14].

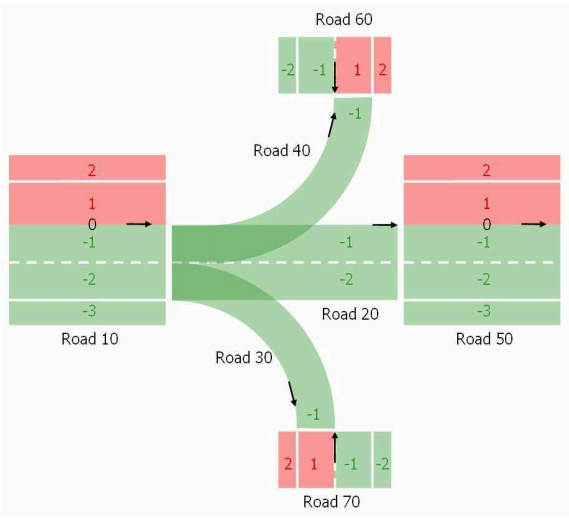


Abbildung 5: Verbindung von Straßen durch einen Junction-Eintrag. Bildquelle: [7, S.76].

Zweitens kann eine Straße in eine Kreuzung (Junction) übergehen, hierbei wird im Predecessor bzw. Succesor-Eintrag ein Junction-Eintrag referenziert. Dies ist nötig wenn die Straße mehr als einen Vorgänger bzw. Nachfolger besitzt. Über eine Liste von Connection-Einträgen innerhalb des Junction-Eintrags wird eine Zuordnung von Straßen getroffen. Dabei geht eine Straße in einen Pfad (Path) über. Mit Pfad werden alle Straßen bezeichnet die sich innerhalb einer Kreuzung befinden. In Abbildung 5 ist die Verbindung über eine Kreuzung dargestellt. Hier kann man sehen, dass die Straße 10 über die Pfade 20, 30 und 40 in die Straßen 50, 60 und 70 übergehen kann. Die drei Straßen 50, 60 und 70 benötigen natürlich eigene Pfade, die wiederum in der Kreuzung referenziert werden müssen.

2.3. Straßenbeschreibung / Trassierung

Zur Beschreibung von Straßenverläufen werden verschiedene geometrische Elemente genutzt. Es hat sich etabliert Geraden, Kreisabschnitte und Klothoide[5] zu verwenden. Eine Straße wird durch die Aneinanderreihung dieser Elemente beschrieben. Interessant ist, dass die Elemente schon bei der Trassierung von Straßen genutzt werden. Für diese Arbeit sind sie relevant, weil sie die Hauptbeschreibungselemente der OpenDRIVE®-Spezifikation sind und dort zur Beschreibung der Straßenreferenzlinien dienen [7, S.13]. Folgend sind die Elemente näher beschrieben.

[5] Auch als Euler- oder Cornu-Spiralen bekannt.

2.3.1. Gerade

Die Gerade[6] ist das einfachste der drei Elemente. Zur genauen Beschreibung einer Geraden werden vier Parameter benötigt, entweder die x- und y-Koordinate des Start- und Endpunkts oder die Koordinaten des Startpunktes sowie die Länge der Strecke und einen Winkel α um die Richtung zu bestimmen. [14]

	Parameter Möglichkeit 1		**Parameter Möglichkeit 2**
1.	`Start_x`	1.	`Start_x`
2.	`Start_y`	2.	`Start_y`
3.	`End_x`	3.	`Länge`
4.	`End_y`	4.	`Alpha`

Tabelle 2: Parameter zur eindeutigen Bestimmung einer Geraden [14].

2.3.2. Kreisabschnitt

Das zweite Element ist der Kreisabschnitt. Es handelt sich hierbei, wie der Name schon sagt, um den Ausschnitt eines Vollkreises. Ein Kreisabschnitt kann durch fünf Parameter genau beschrieben werden. Ein Startpunkt und eine Orientierung (α) bestimmen die Lage des Elements in der Ebene, die Länge bestimmt den Anteil des Kreises und die Krümmung (Kehrwert des Kreisradius) bestimmt wie stark der Abschnitt gekrümmt ist. Durch das Vorzeichen der Krümmung wird außerdem die Richtung des Elementes bestimmt, also ob es mit dem oder gegen den Uhrzeigersinn verläuft. [14]

	Paramerter
1.	`Start_x`
2.	`Start_y`
3.	`Alpha`
4.	`Krümmung (1/radius)`
5.	`Länge`

Tabelle 3: Parameter zur eindeutigen Bestimmung eines Kreisabschnittes [14].

2.3.3. Klothoide (engl. Euler Spiral)

Das letzte der drei Geometrieelemente wird Klothoide genannt. Das Besondere an Klothoiden ist, dass sie eine linear steigende Krümmung besitzen. Durch einen Sprung der Krümmung wäre der Fahrer gezwungen seine Geschwindigkeit zu verringern oder Kurven zu schneiden. So sollte sich zwischen dem Übergang von einer Geraden zu einem Kreisabschnitt und umgekehrt stets eine Klothoide befinden um Sprünge in der Krümmung und dadurch unbequemes Fahrverhalten zu vermeiden. In Abbildung 6 ist der Krümmungsverlauf einer Straße bestehend aus den drei Elementen Klothoide, Grade und Kreisabschnitt dargestellt. [14]

Die Berechnung von Klothoiden ist schwieriger als die von Geraden und Kreisabschnitten. Es werden dazu mindestens sechs Parameter Benötigt. In der folgenden Tabelle ist eine Auflistung dieser zu sehen.

Früher wurden Klothoiden durch das Skalieren einer Einheitsklothoide anhand von in Tabellen vorberechneten Werten bestimmt, was durch die Eigenschaft der Selbstähnlichkeit möglich war.

[6] Eigentlich ist es eine Strecke da eine Gerade keinen Start- und Endpunkt besitzt.

Parameter
1. `Start_x`
2. `Start_y`
3. `Alpha`
4. `Startkrümmung`
5. `Länge`
6. `Endkrümmung`

Tabelle 4: Parameter zur eindeutigen Bestimmung einer Klothoiden [14].

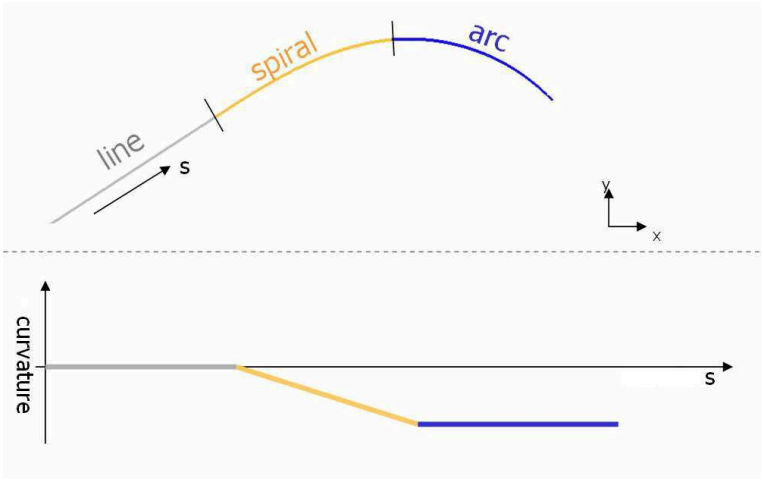


Abbildung 6: Oben: Aneinanderreihung der Geometrieelemente Strecke, Klothoide und Kreisabschnitt wie sie in einer Straße genutzt werden. Unten: Zugehöriger Krümmungsverlauf der Elemente. Abbildung nach [7, S.13].

Durch die heutzutage zur Verfügung stehende Rechenleistung ist es jedoch möglich Klothoiden zur benötigten Zeit mit den nötigen Parametern neu zu berechnen.

$$A^2 = R \cdot L \tag{1}$$

$$T = \frac{L^2}{2A^2} = \frac{L}{2R} \tag{2}$$

$$X = \int_0^L \cos \frac{L^2}{2A^2} \, \mathrm{d}L \tag{3}$$

$$Y = \int_0^L \sin \frac{L^2}{2A^2} \, \mathrm{d}L \tag{4}$$

Zur Berechnung werden Fresnel-Integrale[7] verwendet. Die Formeln 1 bis 4 stellen Formeln zur Berechnung von Klothoiden dar. A ist der sogenannte Klothoidenparameter. Klothoiden mit $A = 1$ werden Einheitsklothoiden genannt. L stellt die Bogenlänge und R den Radius dar. T gibt den Schnittwinkel der Tangenten an. Mit den Formeln (3) und (4) können die Karthesischen Koordinaten (X/Y) für einen bestimmten Punkt auf der Klothoiden berechnet werden. Die Berechnung der Koordinaten geschieht also durch eine Integration von der Länge 0 bis zur Bogenlänge. In Abbildung 7 ist ein Beispiel für eine Klothoide dargestellt.

[7] Benannt nach dem Physiker Augustin Jean Fresnel.

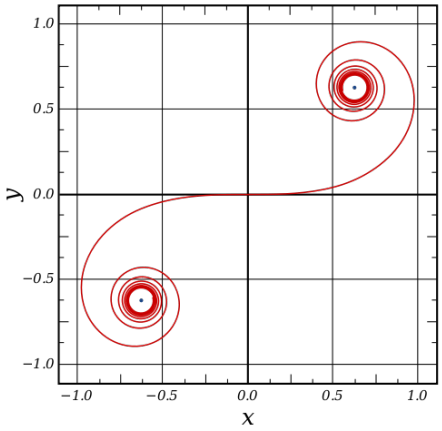


Abbildung 7: Beispielhafte Form einer Klothoide. Bildquelle: `http://de.wikipedia.org/wiki/Klothoide`.

2.4. Verkehrsnetzwerkmodelle für virtuelle Umgebungen

In diesem Abschnitt sollen bestehende Verkehrsnetzwerkmodelle für virtuellen Umgebungen betrachtet werden. Informationen zu diesem Thema sind relativ rar, die meisten Verkehrsnetzwerkmodelle arbeiten mit einer graphähnlichen Repräsentation. In den folgenden zwei Abschnitten werden vier verschiedene in der Literatur zu findende Verkehrsnetzwerkmodelle betrachtet. Zwei davon sind für den Einsatz in Videospielen konzipiert und die anderen beiden stellen Teile von Forschungsprojekten dar.

2.4.1. Verkehrsnetzwerkmodelle in Videospielen

Zu diesem Thema werden zwei Teilgebiete betrachtet die unterschiedliche Ansätze verfolgen. Zuerst wird ein Modell zur Repräsentation von Rennstrecken betrachtet und anschließend das Modell eines Videospiels welches größtenteils urbane Verkehrsnetzwerke beinhaltet.

Repräsentation von Rennstrecken für KI

Das erste betrachtete Verkehrsnetzwerkmodell beschäftigt sich mit der Repräsentation von Rennstrecken für KI-Anwendungen. Es wird im Kapitel *Representing a Racetrack for the AI* des Buchs *AI Game Programming Wisdom* [3] vorgestellt. Jede Rennstrecke besteht aus einer Anzahl von Sektoren welche durch Schnittstellen miteinander verbunden sind. Es ist möglich das Modell auf Basis einer doppelt verketteten Liste zu implementieren. Durch diese Vorgehensweise können unterschiedliche Kurse auf der Strecke realisiert werden (Ideallinie, Überholspur, etc.). Es gibt jedoch mehrere Nachteile die vermuten lassen, dass sich dieses Modell nicht für den Einsatz in unserem System eignet. Zum einen wird es relativ aufwändig sein, Verzweigungsabschnitte in dieses Modell zu integrieren und zum anderen ist es nicht dafür vorgesehen unterschiedliche Arten von Informationen hinterlegen zu können. Die Anforderungen an eine Repräsentation für Rennstrecken unterscheiden sich zu sehr von denen für ein urbanes Verkehrsnetzwerk – wie es in diesem Projekt benötigt wird.

Mafia II Verkehrsnetzwerkmodell

Das zweite betrachtete Verkehrsnetzwerkmodell ist das im Videospiel Mafia II[8] genutzte Modell welches von dem Unternehmen 2K Czech[9] entwickelt wurde. Es liegt die Vermutung nahe, dass in

[8] 2K Games, 2010
[9] `www.2kczech.com`

anderen Computerspielen wie z.B. der Computerspielserie GTA oder dem neu erschienenen SimCity (2013) ähnliche Verkehrsnetzwerkmodelle genutzt werden, zu welchen jedoch auf Grund der generell raren Informationen zu Computerspielen keine bekannten Informationen vorliegen. Informationen zum Mafia II Verkehrsnetzwerkmodell sind in zwei Präsentationen des Mitentwicklers Jan Kratochvíl [15][16] zu finden. Die Handlung des Spiels findet in der fiktiven Metropole Empire Bay statt, welche aus 19 Bezirken besteht und eine ungefähre Größe von 20 km² hat. In den Präsentationen sind ein paar nützliche Informationen zum genutzten Modell enthalten:

- Zur Beschreibung der Verkehrsnetzwerke werden Catmull-Rom-Splines verwendet.
- Das Verkehrsnetzwerk der ganzen Stadt (inklusive aller nötigen Metadaten) ist nur etwa 400Kb groß. Durch diese geringe Größe ist es möglich, das Netzwerk während der Spielsimulation komplett im Speicher zu halten, also auf einem statischen Netzwerk zu arbeiten. Dies hat zur Folge, dass kein dynamisches Nachladen von Netzwerkabschnitten realisiert werden muss.
- Die Navigation auf Kreuzungsbereichen ist ebenfalls durch die Nutzung von Catmull-Rom-Splines realisiert.

Weitere Informationen zum Verkehrsnetzwerkmodell können aus Abbildungen in den beiden Vorträgen gewonnen werden. Dort ist der genutzte Verkehrsnetzwerkeditor dargestellt. Es ist ein erstelltes Verkehrsnetzwerk und Parameterlisten eines Elements zu sehen. Zusätzlich sind Verkehrsnetzwerke während des Spielbetriebs dargestellt. Man kann sehen, dass das Modell aus unterschiedlichen Elementen besteht. Es existieren Elemente für Hauptfahrspuren sowie anliegende Standstreifen. Zusätzlich lassen die Abbildungen auf die existenz weniger priorisierter Nebenstraßen schließen. Jeder Kreuzungsabschnitt enthält eine Menge an Wegen welche unterschiedliche Möglichkeiten darstellen den Kreuzungsabschnitt zu überqueren. Die Flächen der Kreuzungsabschnitte sind durch zylinderartige Elemente abgegrenzt. Außerdem existieren auf den Mittelspuren der Straßen noch graue Würfel, welche so etwas wie Wegpunkte für die Straße darstellen könnten.

2.4.2. Verkehrsnetzwerkmodelle in Forschungsprojekten

Zu diesem Thema wurden Modelle aus zwei Forschungsprojekten betrachtet. Das erste Projekt beschäftigt sich mit der Integration von Verkehrsnetzwerken in bestehende virtuelle Umgebungen und das zweite Projekt beschäftigt sich mit der Generierung von Netzwerken für eine mesoskopische Simulation auf Basis zellulärer Automaten.

Dublin City Verkehrsnetzwerkmodell

Die beiden Publikationen [9] und [10] beschäftigen sich mit einem Verkehrsnetzwerkmodell für virtuelle Umgebungen. Ziel war es in das Dublin City Model, welches im Rahmen des Virtual Dublin Project [11] erstellt wurde, eine Verkehrsnetzwerksrepräsentation zu integrieren ohne dabei das 3D-Modell störend zu beeinflussen. Das Verkehrsnetzwerk soll anschließend fahrzeugsteuernden Agenten als Grundlage für die Navigation durch das Stadtmodell dienen. Das Modell arbeitet mit zwei Komplexitätsstufen innerhalb eines Verkehrsgraphen. Die erste Stufe wird durch einen einfachen durchsuchbaren Graphen dargestellt. Weiterhin sind die einzelnen Straßen in kleinere durchsuchbare Graphen aufgeteilt welche jeweils eine Anzahl adjazenter Knoten enthalten. Dabei entspricht die Größe jedes Knotens der eines durchschnittlichen Fahrzeugs. Durch diese Vorgehensweise wird es ermöglicht auf einfache Weise Stau oder zähfließenden Verkehr zu überwachen. Weiterhin sei es durch dieses Modell möglich auf einfache Weise Fahrzeugfolge- oder Spurwechselmodelle abzubilden. Ein weiterer interessanter Aspekt ist das ein Tool zur schnellen Erstellung von Verkehrsnetzwerken innerhalb einer existierenden 3D-Umgebung (Rapid Construction

of Road Network within an Existing 3D Environment) vorgestellt wird. Dabei wird ein Editor Modul verwendet, welches point-and-click basiert arbeitet.

Ähnliche Tools bestehen bereits im AVeSi Projekt. Jedoch wurde festgestellt, dass die manuelle Erstellung immer noch einen enormen Zeitaufwand bedeutet und die Abbildung realer Gegebenheiten nicht präzise genug geschieht, weshalb eine automatische Generierung angestrebt ist.

Verkehrsnetzwerkmodell für Simulation auf Basis von zellulären Automaten

In [1] wird ein Verkehrsnetzwerkmodell für eine mesoskopische Simulation auf Basis von zellulären Automaten beschrieben. Interessant ist hier, dass die erzeugten Straßengraphen (road graphs), wie auch in diesem Projekt angestrebt, auf Basis realer Daten (real-life data) generiert werden. Als Datenbasis dienen Daten des Projektes OS Land-Line.Plus®[10]. In der Veröffentlichung sind einzelne Aspekte des Modells dargestellt. So werden die in OS Land-Line.Plus® beschriebenen Straßenmittellinien gezeigt sowie auf basis dieser informationen identifizierte Kreuzungen und Straßenelemente. Letztendlich wird die Aufteilung eines Teils des Netzwerks in einzelne Zellen gezeigt. In dem vorgestellten Modell wird zwischen *terminal junctions* an Netzwerkenden und *road junctions* zwischen verschiedenen Straßen unterschieden. Ein Nachteil des Modells ist, dass alle Straßen momentan nur unidirektional betrachtet werden und dass Kreuzungen nur sehr vereinfacht betrachtet werden.

2.5. Simulationsansatz im AVeSi-Projekt

In diesem Abschnitt soll der momentan im AVeSi-Projekt verfolgte Simulationsansatz betrachtet werden. Eines der Ziele des Projektes ist es, Agenten für den Einsatz in virtuellen Verkehrsnetzwerken zu entwickeln. Diese Agenten sollen folgende Eigenschaften erfüllen:

1. Agenten sollen geltende Verkehrsregeln befolgen.

2. Agenten sollen individuelles Verhalten zeigen.

3. Agenten soll es möglich sein Risiken in Kauf zu nehmen und folglich Fehlverhalten zu zeigen.

Einer der Gründe für die Entwicklung ist es, eine realitätsnahe Verkehrssimulation für den Einsatz in virtuellen Umgebungen zu schaffen, welche z.B. im Bereich der Verkehrserziehung (siehe FIVIS-Projekt[11]) eingesetzt werden kann. Um einen solchen Realismus zu erreichen, wird angestrebt, kognitive Eigenschaften zu modellieren. Dazu gehört z.B. die Entwicklung von Persönlichkeiten und Stimmungen für einzelne Verkehrsteilnehmer. Um dies zu erreichen, wurde ein mikroskopischer Simulationsansatz auf Basis eines Multiagentensystems gewählt. Problem dabei ist, dass durch den hohen Rechenaufwand nur vergleichsweise wenige Agenten simuliert werden können. Der verfolgte Ansatz zur Lösung dieses Problems ist es in der unmittelbaren Umgebung eines Akteurs (zumindest in dessen Sichtfeld und etwas darüber hinaus) ein realistisches Verhalten durch einen mikroskopischen Simulationsansatz zu erzeugen. Ab einer gewissen Distanz (die über das Sichtfeld des Akteurs hinausgeht) wird die Simulation in eine, wesentlich effektiver zu berechnende, in diesem Fall warteschlangenbasierte Mesosimulation überführt [5]. Durch den Mesosimulationsansatz sollen die Eigenschaften der einzelnen Agenten persistent in der Simulationsumgebung erhalten bleiben, seien es physikalische Merkmale wie Autofarbe, Autotyp oder sich während der Mikrosimulation entwickelnde Emotionsprofile.

[10] `http://edina.ed.ac.uk/digimap/description/products/landline.shtml`
[11] `http://vc.inf.h-bonn-rhein-sieg.de/?page_id=425`

Verkehrsnetzwerkmodell für die mesoskopische Simulationsebene

Das Konzept für die mesoskopische Simulationsebene wurde im Rahmen der Masterarbeit von T. Dettmar (*Queuing Models for Traffic Simulations in Virtual Environments*) [5] entwickelt. Das Modell arbeitet auf Basis von Warteschlangen und basiert auf dem FastLane Modell welches in der Doktorarbeit von C. Gawron [8] eingeführt wurde. Die Hauptbestandteile des Modells sind Knoten und Kanten, durch welche eine graphähnliche Darstellung des Verkehrsnetzwerks entsteht. Wichtig ist hier noch zu erwähnen, dass im Gegensatz zum mikroskopischen Simulationsansatz hier die Netzwerkelemente die aktive Rolle übernehmen. Das heißt, dass die Knoten und Kanten die eigentlichen Agenten auf dieser Ebene darstellen. Die Verkehrsteilnehmer werden passiv, in Form von Paketen, durch das Netzwerk bewegt. Im Gegensatz zum Modell für die mikroskopische Ebene findet in mehreren Punkten eine Abstraktion statt. Fahrspuren werden komplett abstrahiert. Die die Straße repräsentierenden Kanten arbeiten hauptsächlich auf Basis einer Längenangabe. Diese reicht aus, um die Fahrzeit auf der Kante für die Verkehrsteilnehmer zu berechnen. Die Repräsentation von Kreuzungen geschieht ausschließlich durch Knotenelemente, welche die Verbindungsinformationen sowie Vorfahrtsprioritäten enthalten. Daraus folgt, dass die Pfade auf den Kreuzungen – welche ein wichtiger Bestandteil im Modell der mikroskopischen Ebene sind – von diesem Modell nicht betrachtet werden. Die größte Abstraktion geschieht dadurch, dass die Geometrie des Straßennetzes quasi komplett vernachlässigt wird. Für den Simulationsprozess sind außer der Länge der Kanten und der Verbindungen zwischen den Kanten keine Geometrieinformationen nötig. Jedoch sind Geometrieinformationen für eine Visualisierung des Modells, welche zumindest während der Debugphasen benötigt wird nötig. Zusätzliche Detailinformationen, welche für mikroskopische Simulationsansätze in Betracht gezogen werden könnten, sind für das Modell nicht von Bedeutung.

Verkehrsnetzwerkmodell für die mikroskopische Simulationsebene

Im Vorfeld zu diesem Projekt bestand kein definiertes Verkehrsnetzwerkmodell für die mikroskopische Simulationsebene. Jedoch existierten bereits einige unstrukturierte Realisierungen. Dies stellte ein Problem dar, welches einer der ausschlaggebenden Punkte für dieses Projekt war. Um Unklarheiten in folgenden Projekten zu beseitigen und künftige Arbeiten auf einer soliden Basis beginnen zu können bestand dringender Bedarf nach einer Formalisierung des Modells. Bei der Formalisierung sollte darauf geachtet werden, bestehende Sachverhalte zu berücksichtigen und so keine vollkommene Überarbeitung bestehender und momentaner Arbeiten notwendig zu machen, sondern diese lediglich anpassen zu müssen.

3. Semantische Verkehrsnetzwerkmodelle

In diesem Abschnitt wird auf die Verkehrsnetzwerkmodelle für die beiden Simulationsebenen eingegangen. Folgend existiert für jede Simulationsebene ein Unterabschnitt (3.1 und 3.2). In diesen wird jeweils zuerst auf das Modell eingegangen und anschließend diskutiert, welche Daten von den Modellen benötigt werden und an welchen Stellen innerhalb der OpenDRIVE®-Spezifikation entsprechende Daten gewonnen werden können.

3.1. Mesoskopische Simulationsebene

Das Verkehrsnetzwerkmodell für die mesoskopische Simulationsebene – dessen Entwicklung Teil einer anderen Arbeit war – wurde bereits in Abschnitt 2.5 eingeführt. Im folgenden Abschnitt werden die Informationen dargestellt, welche durch das Verkehrsnetzwerkmodell für die Simulation bereitgestellt werden müssen. Anschließend wird diskutiert an welchen Stellen innerhalb der OpenDRIVE®-Spezifikation entsprechende Informationen gewonnen werden können. Der umgesetzte Überführungsprozess der OpenDRIVE®-Daten in das mikroskopische Modell wird in Abschnitt 4.4.2 betrachtet.

Informationen und OpenDRIVE®

In diesem Abschnitt wird betrachtet welche Informationen vom mesoskopischen Modell benötigt werden. Anschließend wird analysiert ob und wenn ja an welcher Stelle der OpenDRIVE®-Beschreibung diese Informationen gewonnen werden können. In den Tabellen 5 und 6 sind die von den jeweiligen Modellelementen benötigten Informationen zusammengefasst dargestellt. Folgend muss beachtet werden, dass zwischen zwei Arten von Informationen unterschieden werden muss. Erstere sind diejenigen Informationen die für Berechnungen während des Simulationsvorgangs benötigt werden. Zweitere können ergänzend zu Visualisierungs- oder Debugzwecken genutzt werden.

Informationen für Kanten

Innerhalb der Kanten des Graphen werden folgende Informationen benötigt: Die Länge des repräsentierten Straßenabschnitts, der Startknoten und der Endknoten, die durchschnittliche zugelassene Geschwindigkeit auf dem repräsentierten Straßenabschnitt und die maximale Verkehrsdichte. Die maximale Verkehrsdichte ϱ_{max} kann über die Straßenlänge L und die Anzahl der Spuren der Straße n_{lanes} anhand von Formel 5 bestimmt werden. Die Variable l entspricht der durchschnittlichen Platzlänge die ein Fahrzeug im Fall einer Stausituation einnimmt. Es kann ein konstanter Wert eingesetzt werden. Werte hierfür sind in entsprechender Literatur zu finden, z.B. [18] oder [4].

$$\varrho_{max} = \frac{L}{l} \cdot n_{lanes} \tag{5}$$

Zur Visualisierung der Kante wird zusätzlich die Position des Startknotens sowie die Position des Endknotens der Kante benötigt. Die zur Generierung der Kanten benötigten Informationen können größtenteils aus den *Road Records* der OpenDRIVE®-Spezifikation gewonnen werden. Eine zusammenfassende Liste mit den benötigten Parametern ist in Tabelle 5 zu finden.

1. Der Parameter für die ID einer Straße kann direkt aus dem *Road Header Record* entnommen werden, wo für jede Straße eine eindeutige ID vergeben wird.

	Simulation
1.	Straßen ID
2.	Straßenabschnittslänge
3.	Startknoten
4.	Endknoten
5.	Durchschnittliche zugelassene Geschwindigkeit
6.	Maximale Verkehrsdichte
	Visualisierung
7.	Position des Startknotens
8.	Position des Endknotens

Tabelle 5: Informationen für Kanten.

2. Die Länge des Straßenabschnitts kann ebenfalls direkt aus dem *Road Header Record* entnommen werden. Hier kann der Wert für die Länge der Straßenreferenzlinie verwendet werden. Die Länge der anliegenden Spuren wird zwar ein wenig von der Länge der Straßenreferenzlinie abweichen, dies kann jedoch auf Grund der Abstraktion durch die mesoskopischen Ebene vernachlässigt werden.

3. Der Startknoten kann aus dem *Road Predecessor* Element innerhalb des *Road Link Record* ermittelt werden. An dieser Stelle ist die ID der Vorgängerstraße hinterlegt. Über diese ID kann der zugehörige Knoten ermittelt werden.

4. Der Endknoten kann aus dem *Road Successor* Element innerhalb des *Road Link Record* ermittelt werden. An dieser Stelle ist die ID der Nachfolgerstraße hinterlegt. Über diese ID kann der zugehörige Knoten ermittelt werden.

5. Die durchschnittlich zugelassene Geschwindigkeit kann aus dem *Road Lane Speed Record* innerhalb des *Lane Record* ermittelt werden. An dieser Stelle muss eine Mittlung stattfinden. Dabei müssen mehrere Aspekte berücksichtigt werden. Innerhalb eines *Lane Record*-Eintrags können sich mehrere *Lane Speed Record* Einträge befinden, die enthaltenen Geschwindigkeiten müssen unter Berücksichtigung der Länge ihres Gültigkeitsbereiches gemittelt werden. Zudem kann es vorkommen das mehrere benachbarte Lanes definiert sind. Wenn dies der Fall ist muss wiederum eine Mittlung über die Durchschnittsgeschwindigkeiten der einzelnen *Lane Record* Einträge stattfinden. Letztendlich ist es möglich das sich eine Kante über mehrere *Road Lane Section Record* Einträge erstreckt. Um die endgültige Durchschnittsgeschwindigkeit zu berechnen muss wiederum eine Mittlung über die Durchschnittsgeschwindigkeiten der *Road Lane Section Record* Einträge erfolgen, dabei müssen deren Längen berücksichtigt werden.

6. Um die maximale Verkehrsdichte zu ermitteln werden nach Formel 5 die Straßenabschnittslänge, die Anzahl der Spuren der Straße und eine durchschnittliche Fahrzeuglänge benötigt. Die durchschnittliche Fahrzeuglänge ist ein konstanter Parameter und kann wie oben schon erwähnt entsprechender Literatur entnommen werden. Die Ermittlung der Straßenabschnittslänge ist schon in Punkt 2 betrachtet worden. Übrig bleibt noch die Anzahl der Spuren der Straße. Diese kann ermittelt werden in dem die *Lane Record* Einträge der entsprechenden Straßenseite innerhalb des *Road Lane Section Record* gezählt werden. Wenn sich der Straßenabschnitt über mehrere *Road Lane Section Record* Einträge erstreckt, muss die Anzahl der Spuren über diese unter Berücksichtigung derer Länge gemittelt werden.

7. Die Position des Startknotens kann über den durch Parameter 3 referenzierten Startknoten ermittelt werden. Darauf, wie die Position eines Knotens bestimmt werden kann, wird im

	Simulation
1.	Liste mit Vorgängerkanten
2.	Liste mit Nachfolgerkanten
3.	Übergangswahrscheinlichkeiten
	Visualisierung
4.	Position des Knotens

Tabelle 6: Informationen für Knoten.

folgenden Abschnitt (Punkt 4) eingegangen.

8. Die Position des Endknotens kann über den durch Parameter 4 referenzierten Endknoten ermittelt werden. Darauf, wie die Position eines Knotens bestimmt werden kann, wird im folgenden Abschnitt (Punkt 4) eingegangen.

Informationen für Knoten

Innerhalb der Knoten des Graphen werden folgende Informationen benötigt: Als erstes eine Liste mit Vorgänger- sowie eine Liste mit Nachfolgerkanten. Dazu eine Liste mit Übergangswahrscheinlichkeiten zu den entsprechenden Nachfolgerkanten und zur Visualisierung eine Position für den Knoten. Die zur Generierung der Knoten benötigten Informationen werden hauptsächlich aus dem *Junction Record* der OpenDRIVE®-Spezifikation gewonnen. Eine zusammenfassende Liste der Parameter ist in Tabelle 6 zu finden.

1. Die Liste der Vorgängerkanten kann über die im *Junction Record* referenzierten *incommingRoads* ermittelt werden. Die IDs der *incommingRoads* entsprechen den Vorgängerkanten.

2. Die Liste der Nachfolgerkanten kann über die im *Junction Record* referenzierten *connectingRoads* ermittelt werden. Da die *connectingRoads* jedoch nur den verbindenden Pfad darstellen, muss zuerst die entsprechende *connectingRoad* ermittelt werden, anschließend kann über den *Road Successor*-Eintrag der *connectingRoad* die nachfolgende Straße bestimmt werden. Die ID dieser entspricht der der Nachfolgerkante.

3. Um Übergangswahrscheinlichkeiten für die entsprechenden eingehenden bzw. ausgehenden Kanten eines Knotens zu ermitteln bestehen mehrere Möglichkeiten.

   a) Die erste Möglichkeit ist es die Daten aus dem *Junction Priority Record* auszuwerten. An dieser Stelle wird eine Anzahl von Einträgen hinterlegt. Jeder Eintrag enthält die IDs zweier Pfade, dabei ist ein Pfad als *high* und der andere Pfad als *low* hinterlegt. Verkehrsteilnehmer auf dem *high*-Pfad haben Vorfahrt und Verkehrsteilnehmer auf dem *low*-Pfad müssen warten wenn ein Verkehrsteilnehmer den *high*-Pfad beansprucht.

   b) Die zweite Möglichkeit ist es Informationen aus dem *Road Signal Record* auszuwerten. An dieser Stelle sind Informationen zu Verkehrszeichen und Signalen hinterlegt. Diese Signale können zusätzlich durch Informationen innerhalb des *Controller Record* und des *Junction Controller Record* koordiniert werden. Der *Controller Record* liefert eine Anzahl konsistenter Zustände für eine Gruppe von Signalen. Der *Junction Controller Record* enthält Verweise auf die Controller-Einträge welche für die Koordination eines Junction-Eintrags genutzt werden.

An dieser Stelle sollte beachtet werden, dass die Auswertung der in Möglichkeit (b) beschriebenen Daten wesentlich schwieriger ist. Gründe dafür sind, dass die Einträge zu Verkehrszeichen bzw. Signalen keinen direkten Bezug zu einem Junction-Eintrag haben müssen, welcher evtl. zusätzlich hergestellt werden muss. Signale können auch dynamische Zustände beschreiben, was ein dynamisches Ändern der Prioritäten oder eine statistische Auswertung der Schaltzeiten der Signale nötig macht.

4. Da der *Junction Record* an sich keinen Parameter für die Position eines Junction-Eintrags besitzt, muss diese auf eine andere andere Weise ermittelt werden. Eine Möglichkeit ist es einen Mittelwert aus den Startpunkten der im *Connection Record* referenzierten Path Elemente – also einen Mittelwert aus den Startpunkten der Pfade auf der Kreuzung – zu bilden. Die Startpunkte sind direkt im *Road Plan View Record* des *Road Records* zu finden, müssen also nicht extra berechnet werden.

3.2. Mikroskopische Simulationsebene

Die mikroskopische Simulationsebene und das dazugehörige Modell stellen die Grundlage für die Entwicklung kognitiver Agenten dar. Um die bereits in Abschnitt 2.5 beschriebenen Ziele erfüllen zu können, muss das unterliegende Verkehrsnetzwerkmodell wesentlich detaillierter sein als das vorgestellte Modell für die mesoskopische Ebene. Im folgenden Abschnitt wird das Modell formal eingeführt. Anschließend werden die für die Simulation benötigten Informationen dargestellt und diskutiert, an welchen Stellen innerhalb der OpenDRIVE®-Spezifikation, entsprechende Informationen gewonnen werden können. Der umgesetzte Überführungsprozess der OpenDRIVE®-Daten in das mikroskopische Modell wird in Abschnitt 4.4.2 betrachtet.

3.2.1. Verkehrsnetzwerkmodell für die mikroskopische Simulationsebene

Im Folgenden wird das Verkehrsnetzwerkmodell für die mikroskopische Simulationsebene formal beschrieben. Dazu werden die Elemente des Modells definiert und die Beziehungen zwischen diesen beschrieben.

L ist dabei eine Menge von Spuren (Lanes). Jede Spur beschreibt ein Straßensegment zwischen zwei Verzweigungen. C ist eine Menge von Connectoren. Jeder Connector beginnt an einer Stelle, an der eine Spur in eine Verzweigung übergeht. Der Connector führt durch diese Verzweigung (meistens eine Kreuzung) und führt letztendlich zu einer weiteren, von der Verzweigung wegführenden, Spur. Ein Connector beschreibt also einen möglichen Weg innerhalb einer Verzweigung.

$$L = \{l_1, l_2, \ldots, l_n\}, n \in \mathbb{N},$$

$$C = \{c_1, c_2, \ldots, c_m\}, m \in \mathbb{N}$$

R ist eine Menge von Straßen (Roads). Eine Straße ist ein Element, welches eine Menge von Spuren zusammenfasst. Dazu werden die beiden Funktionen $road$ und $lanes$ definiert. Die Funktion $road$ ordnet jeder Spur eine Straße zu. Die Funktion $lanes$ ordnet einer Straße eine einzigartige Teilmenge aller Spuren zu.

$$R = \{r_1, r_2, \ldots, r_o\}, o \in \mathbb{N},$$

$$road : L \to R,$$

$$lanes : R \to \wp(L),$$

$$lanes(r) = \{l \in L \mid road(l) = r\},$$

$$l \in lanes(r_i) \wedge l \in lanes(r_j) \Rightarrow r_i = r_j, i, j \in \mathbb{N}$$

P ist eine Menge von Pfaden (Paths). Ein Pfad ist ein Element, welches eine Menge von Connectoren zusammenfasst. Dazu werden die beiden Funktionen $path$ und $connectors$ definiert. Die

Funktion $path$ ordnet jedem Connector einen Pfad zu. Die Funktion $connectors$ ordnet einem Pfad eine einzigartige Teilmenge aller Connectoren zu.

$$P = \{p_1, p_2, \ldots, p_q\}, q \in \mathbb{N},$$

$$path : C \to P,$$

$$connectors : P \to \wp(C),$$

$$connectors(p) = \{c \in C \mid path(c) = p\},$$

$$c \in connectors(p_i) \wedge c \in connectors(p_j) \Rightarrow p_i = p_j, i, j \in \mathbb{N}$$

Um die Vorgänger- und Nachfolgerbeziehungen der Spuren sowie Connectoren zu beschreiben, werden die folgenden Funktionen eingeführt. Jeder Connector besitzt eine eindeutige Vorgängerspur sowie eine eindeutige Nachfolgerspur $(1 : 1)$. Jede Spur hingegen besitzt eine Menge an vorangehenden sowie eine Menge an nachfolgenden Connectoren $(1 : n/1 : m)$.

Die Funktionen $prelane$ und $suclane$ werden definiert, um einem Connector jeweils eine vorangehende und eine nachfolgende Spur zuzuweisen.

$$prelane : C \to L,$$

$$suclane : C \to L$$

Die Funktionen $preconns$ und $succonns$ werden definiert, um einer Spur jeweils eine Menge an vorangehenden und eine Menge an nachfolgenden Connectoren zuzuweisen.

$$preconns : L \to \wp(C),$$

$$preconns(l) = \{c \in C | suc(c) = l\},$$

$$succonns : L \to \wp(C),$$

$$succonns(l) = \{c \in C | pre(c) = l\}$$

In den Abbildungen 8, 9 und 10 sind zusätzlich zu den formalen Beschreibungen erläuternde grafische Darstellungen der eingeführten Elemente vorhanden. Durch die bisher eingeführten Komponenten und deren Beziehungen ist es möglich ein einfaches Netzwerk aufzubauen. Um bisher fehlende geometrische Beschreibungen zum Netzwerkmodell hinzuzufügen, werden Folgen von Wegpunkten definiert. Anschließend bekommt jedes Segment (Spur oder Connector) eine dieser Folgen zugewiesen. Zusätzlich wird jedem Wegpunkt eine Menge an Informationen zugewiesen. Diese kann während des Simulationsprozesses von den Agenten genutzt werden.

W ist eine Menge von Wegpunkten und S ist eine Menge von möglichen Folgen aus diesen Wegpunkten. Die Funktion $waypoints$ ist definiert um jeder Spur und jedem Connector eine dieser Wegpunktsequenzen aus S zuzuweisen.

$$W = \{w_1, w_2, \ldots, w_s\}, s \in \mathbb{N},$$

$$S = \{(w_u, \ldots, w_v \in W, u, v \in \mathbb{N}\},$$

$$waypoints : C \cup L \to S$$

Zudem wird die Funktion pos definiert, um jedem Wegpunkt zwei Koordinaten (x,y) zuzuweisen und ihm somit eine Position in der Ebene zu geben.

$$pos : W \to \mathbb{R}^2,$$

$$pos(w) = (x, y)$$

I ist eine Menge an Informationen. Die Funktion $wpinf$ ist definiert, um jedem Wegpunkt aus W eine Teilmenge der Informationen in I zuweisen zu können.

$$I = \{i_1, i_2, \ldots, i_t\}, t \in \mathbb{N},$$

$$wpinf : W \to \wp(I)$$

Zusätzlich muss eine Möglichkeit geschaffen werden um die Vorfahrtsprioritäten der Segmente zu beschreiben. Dazu wurde eine Menge von Verzweigungselementen (Junctions) definiert. Jedes Verzweigungselement bekommt eine Menge an Informationen zugewiesen, wodurch die Vorfahrtsprioritäten abgebildet werden können und außerdem die Möglichkeit besteht zusätzliche Informationen zu hinterlegen. Weil die Connectoren die möglichen Wege durch die Verzweigungselemente darstellen, wird jeder Connector einem Verzweigungselement zugewiesen. Damit haben die, sich auf einer Spur bewegenden Agenten, die Möglichkeit sich einen möglichen folgenden Connector und dessen zugehöriges Verzweigungselement anzuschauen und die darin enthaltenen Vorfahrtsprioritäten zu ermitteln.

J ist eine Menge von Verzweigungselementen (Junctions). Die Funktion $junction$ wird definiert, um jedem Connector ein Verzweigungselement zuzuweisen und die Funktion $jconns$ wird definiert, um jedem Verzweigungselement eine einzigartige Menge an Connectoren zuzuweisen. $juncinf$ ist eine Funktion die definiert wird, um jedem Verzweigunselement eine Teilmenge der Informationen in I zuweisen zu können.

$$J = \{j_1, j_2, \ldots, j_u\}, u \in \mathbb{N},$$

$$junction : C \to J,$$

$$jconns : J \to \wp(C),$$

$$jconns(j) = \{c \in C \mid junctions(c) = j\},$$

$$c \in jconns(j_k) \wedge c \in jconns(j_l) \Rightarrow j_k = j_l, k, l \in \mathbb{N},$$

$$juncinf : j \to \wp(I)$$

Durch die Definition der vorangehend eingeführten Elemente und derer Beziehungen ist eine graphähnliche Repräsentation von Verkehrsnetzwerken und derer Charakteristiken möglich. Durch das Zuweisen von Wegpunktfolgen zu jeder Spur und jedem Connector ist das Netzwerk außerdem räumlich repräsentiert. Informationen können zu jedem Wegpunkt und dadurch, zu beliebigen Positionen im Netzwerk hinzugefügt werden. Zusätzlich können Informationen zu den Verzweigungselementen hinzugefügt werden, wodurch es möglich wird z.B. Vorfahrtsprioritäten zu hinterlegen. In Abbildung 11 ist eine erläuternde grafische Darstellung zum Konzept der Verzweigungselemente zu sehen.

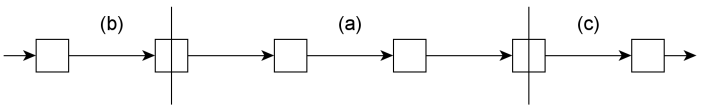


Abbildung 8: Konzept eines Connector-Elements. Ein Connector-Element (a) besteht aus einer Folge von Wegpunkten. Zusätzlich besitzt es einen Verweis auf ein vorangehendes Lane-Element (b) und eine nachfolgendes Lane-Element (c).

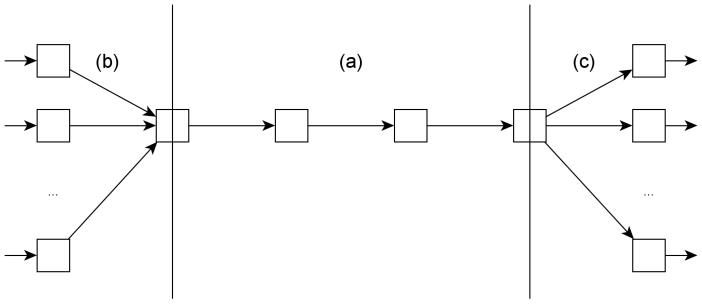


Abbildung 9: Konzept eines Lane-Elements. Eine Lane-Element (a) besteht aus einer Folge von Wegpunkten. Zusätzlich besitzt es eine Menge an vorangehenden Connector-Elementen (b) und eine Menge von nachfolgenden Connector-Elementen (c).

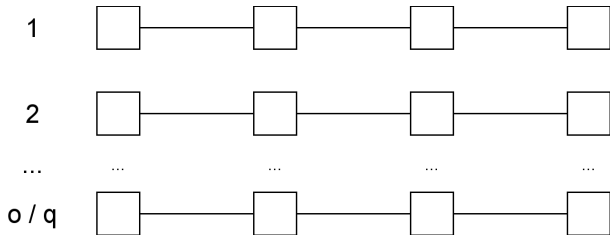


Abbildung 10: Konzept eines Road- bzw. Path-Elements. Ein Road- oder ein Path-Element besteht aus einer Menge von benachbarten Lane- bzw. Connector-Elementen. Innerhalb eines Road- bzw. eines Path-Elements werden benachbarte Lane- bzw. Connector-Elemente zu einem größeren Ganzen zusammengefasst.

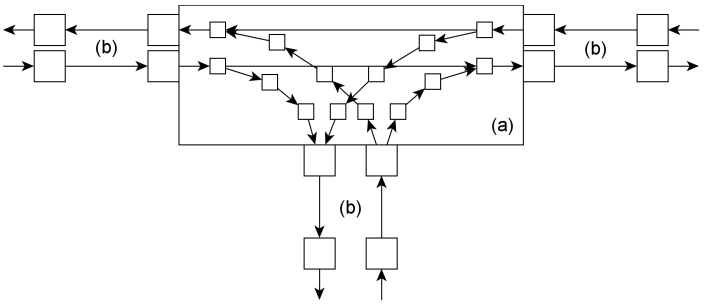


Abbildung 11: Konzept eines Junction-Elements. Ein Junction-Element (a) besteht aus einer Menge von Connector-Elementen. Im Junction-Element können Informationen über die Priorität der Vorgänger- bzw. Nachfolgerelemente (b) der Connector-Elemente hinterlegt werden.

3.2.2. Informationen und OpenDRIVE®

In diesem Abschnitt wird betrachtet welche Informationen vom mikroskopischen Modell benötigt werden und ob und wenn ja an welcher Stelle der OpenDRIVE®-Beschreibung diese Informationen gewonnen werden können. In den Tabellen 7 bis 11 sind die von den jeweiligen Modellelementen benötigten Informationen zusammengefasst dargestellt. Hier wird zwischen primären, also notwendigen Informationen und optionalen Informationen unterschieden.

Informationen für Lane-Elemente

Ein Lane-Element benötigt eine Liste mit vorangehenden und eine Liste mit nachfolgenden Connector-Elementen. Zusätzlich sollen Informationen wie maximale Geschwindigkeit, Breite, Länge

	Primär
1.	Liste mit vorangehenden Connector-Elementen
2.	Liste mit nachfolgenden Connector-Elementen
3.	Maximale Geschwindigkeit
4.	Breite
5.	Länge
6.	Rechte Nachbarspur
7.	Linke Nachbarspur
8.	Liste mit Waypoint-Elementen
	Optional
9.	Spurtyp

Tabelle 7: Informationen für Lane-Elemente.

sowie Informationen zu den Nachbarspuren hinterlegt sein.

1. *Liste mit vorangehenden Connector-Elementen*
   Nur Lane-Elemente deren Startpunkt sich auch am Beginn des zugehörigen Road-Elements befindet, können vorangehende Connector-Elemente besitzen. Lane-Elemente deren Startpunkt sich mitten auf einem Road-Element befindet besitzen kein Vorgängerelement. Dies ist z.B. beim Beginn einer neuen Spur der Fall. Es ist möglich die vorangehenden Connector-Elemente direkt aus der OpenDRIVE®-Beschreibung zu ermitteln, jedoch ist es einfacher diese bei der Ermittlung des nachfolgenden Lane-Elements (siehe Informationen für Connector-Elemente) mit zu erfassen.

2. *Liste mit nachfolgenden Connector-Elementen*
   Nur Lane-Elemente welche mit dem zugehörigen Road-Element enden, können nachfolgende Connector-Elemente besitzen. Lane-Elemente die mitten auf einem Road-Element enden, besitzen keine Nachfolgeelemente. Es ist möglich die nachfolgenden Connector-Elemente direkt aus der OpenDRIVE®-Beschreibung zu ermitteln, jedoch ist es einfacher diese bei der Ermittlung des nachfolgenden Lane-Elements (siehe Informationen für Connector-Elemente) mit zu erfassen.

3. *Maximale Geschwindigkeit*
   Da sich die maximal zugelassene Geschwindigkeit eines Lane-Elements innerhalb des Elements ändern kann, ist es sinnvoll diese nicht im Lane-Element selber, sondern in entsprechenden Waypoint-Elementen (siehe Parameter 8) zu hinterlegen.

4. *Breite*
   Da sich die Breite eines Lane-Elements innerhalb des Elements ändern kann ist es sinnvoll diese nicht im Lane-Element selber, sondern in entsprechenden Waypoint-Elementen (siehe Parameter 8) zu hinterlegen.

5. *Länge*
   Da die Geometrie des Lane-Elements über eine Liste von Waypoint-Elementen (siehe Parameter 8) gegeben ist, kann die Länge eines Lane-Elements über das aufsummieren der Distanz zwischen diesen Waypoint-Elementen berechnet werden.

6. *Rechte Nachbarspur*
   Da sich die rechte Nachbarspur eines Lane-Elements im Verlauf des Elements ändern kann, ist es sinnvoll diese nicht im Lane-Element selber, sondern in entsprechenden Waypoint-

Elementen (siehe Parameter 8) zu hinterlegen.

7. *Linke Nachbarspur*
   Da sich die linke Nachbarspur eines Lane-Elements im Verlauf des Elements ändern kann, ist es sinnvoll diese nicht im Lane-Element selber, sondern in entsprechenden Waypoint-Elementen (siehe Parameter 8) zu hinterlegen.

8. *Liste mit Waypoint-Elementen*
   Zu jedem Lane-Element wird eine Liste von mindestens zwei Waypoint-Elementen benötigt. Durch diese ist es möglich die Geometrie des Elements sowie weitere Informationen, die sich im Laufe des Elements ändern können, darzustellen. Positionen für mögliche Waypoint-Elemente auf einem Lane-Element können wie folgt bestimmt werden: Zuerst muss ein Punkt auf der Referenzlinie eines Road-Eintrags ($\vec{r}$) berechnet werden (näheres dazu in Abschnitt 2.3) und zusätzlich der Richtungsvektor ($\vec{d}$) der Referenzlinie an diesem Punkt. Alle nötigen Informationen für diese Berechnungen finden sich im *Road Plan View Record*. Der Richtungsvektor muss um 90° bzw. 270° gedreht werden damit er orthogonal zur Tangente am Punkt der Referenzlinie steht und somit in Richtung der linken bzw. rechten Spuren weist. Dieser Vektor muss normiert und anschließend mit der Summe der Breiten (lw_i) der inneren Spuren plus der halben Breite der aktuellen Spur (lw_n) multipliziert werden, um ihn auf die entsprechende Länge zu bringen. Informationen zur Breite der Spuren sind im *Lane Width Record* zu finden. Letztlich muss der so entstandene Vektor auf den Vektor des Punktes auf der Referenzlinie ($\vec{r}$) aufaddiert werden, um so den Wegpunktvektor ($\vec{w}$) zu erhalten. Das gerade Beschriebene ist in den Formeln 6 und 7 formal dargestellt.

$$\vec{w} = \vec{r} + \left(\left(\vec{d} \cdot R_{90^\circ} \right) \cdot \left(\sum_{i=1}^{n-1} lw_i + \frac{lw_n}{2} \right) \right) \tag{6}$$

$$\vec{w} = \vec{r} + \left(\left(\vec{d} \cdot R_{270^\circ} \right) \cdot \left(\sum_{i=1}^{n-1} lw_i + \frac{lw_n}{2} \right) \right) \tag{7}$$

   Durch das Anhängen von Informationen an die Waypoint-Elemente ist es möglich Informationen zu den Parametern 3,4,6 und 7 zu hinterlegen.

9. *Spurtyp*
   Zusätzlich ist es möglich Informationen zum Spurtyp innerhalb der Lane-Elemente zu hinterlegen. Innerhalb des *Lane Record* ist ein entsprechendes Attribut vorhanden.

Informationen für Connector-Elemente

Ein Connector-Element benötigt ein vorangehendes sowie ein nachfolgendes Lane-Element. Diese Verbindung ist im Gegensatz zur Lane-Connector Verbindung eindeutig. Zusätzlich werden folgende Parameter benötigt: Maximal zugelassene Geschwindigkeit, Breite, Länge, eine Liste mit Waypoint-Elementen, ein Junction-Element sowie optional ein Spurtyp.

1. *Vorangehendes Lane-Element*
   Um das vorangehende Lane-Element zu ermitteln muss zuerst das vorangehende Road-Element ermittelt werden, dies kann durch den *Road Predecessor*-Eintrag innerhalb der *Road Records* ermittelt werden. Nun muss die id des vorangehenden Lane-Eintrags ermittelt werden. Diese ist im *Lane Link Record* innerhalb des *Lane Record* zu finden. Über diese Informationen kann das vorangehende Lane-Element bestimmt werden. Zusätzlich kann das aktuelle Connector-Element in die Liste mit nachfolgenden Connector-Elementen des Lane-Elements eingetragen werden.

	Primär
1.	Vorangehendes Lane-Element
2.	Nachfolgendes Lane-Element
3.	Maximale Geschwindigkeit
4.	Breite
5.	Länge
6.	Liste mit Waypoint-Elementen
7.	Junction-Element
	Optional
8.	Spurtyp

Tabelle 8: Informationen für Connector-Elemente.

2. *Nachfolgendes Lane-Element*
   Um das nachfolgende Lane-Element zu ermitteln muss zuerst das nachfolgende Road-Element ermittelt werden, dies kann durch den *Road Successor*-Eintrag innerhalb der *Road Records* ermittelt werden. Nun muss die id des nachfolgenden Lane-Eintrags ermittelt werden. Diese ist im *Lane Link Record* innerhalb des *Lane Record* zu finden. Über diese Informationen kann das nachfolgende Lane-Element bestimmt werden. Zusätzlich kann das aktuelle Connector-Element in die Liste mit vorangehenden Connector-Elementen des Lane-Elements eingetragen werden.

3. *Maximale Geschwindigkeit*
   Da sich die maximal zugelassene Geschwindigkeit innerhalb eines Connector-Elements ändern kann ist es sinnvoll diese nicht im Connector-Element selbst, sondern in entsprechenden Waypoint-Elementen (siehe Parameter 6) zu hinterlegen.

4. *Breite*
   Da sich die Breite eines Connector-Elements innerhalb des Elements ändern kann ist es sinnvoll diese nicht im Connector-Element selbst, sondern in entsprechenden Waypoint-Elementen (siehe Parameter 6) zu hinterlegen.

5. *Länge*
   Da die Geometrie des Connector-Elements über eine Liste von Waypoint-Elementen (siehe Parameter 6) gegeben ist kann die Länge eines Connector-Elements über das aufsummieren der Distanz zwischen diesen Waypoint-Elementen berechnet werden.

6. *Liste mit Waypoint-Elementen*
   Die Berechnung von Waypoint-Elementen für Connector-Elemente erfolgt äquivalent zu der für Lane-Elemente (siehe Informationen für Lane-Elemente Parameter 8).

7. *Junction-Element*
   Das zum Connector-Element gehörige Junction-Element kann über eine im *Road Header Record* angegebene junction-id ermittelt werden.

8. *Spurtyp*
   Zusätzlich ist es möglich in Connector-Elementen Informationen zum Spurtyp zu hinterlegen. Im *Lane Record* ist ein entsprechendes Attribut zu finden.

	Primär
1.	Liste mit linken Lane-/Connector-Elementen
2.	Liste mit rechten Lane-/Connector-Elementen
	Optional
3.	id
4.	Länge
5.	Name

Tabelle 9: Informationen für Road- bzw. Path-Elemente.

Informationen für Road- bzw. Path-Elemente

Da Road- und Path-Elemente fast äquivalent sind werden deren Informationen gemeinsam innerhalb eines Abschnitts betrachtet. Jedes Road- und jedes Path-Element benötigt folgende Informationen: Eine Liste mit linken Lane- bzw. Connector-Elementen, eine Liste mit rechten Lane- bzw. Connector-Elementen, sowie optional id, Länge und Name.

1. *Liste mit linken Lane-/Connector-Elementen*
   Innerhalb der *Left Records* sind alle Informationen vorhanden um eine Liste mit linken Lane- bzw. Connector-Elementen aufzubauen. Die zu den in den *Left Records* befindlichen *Lane Record*-Einträgen gehörenden Lane- bzw. Connector-Elemente gehören zur Liste.

2. *Liste mit rechten Lane-/Connector-Elementen*
   Innerhalb der *Right Records* sind alle Informationen vorhanden um eine Liste mit rechten Lane- bzw. Connector-Elementen aufzubauen. Die zu den in den *Right Records* befindlichen *Lane Record*-Einträgen gehörenden Lane- bzw. Connector-Elemente gehören zur Liste.

3. *id*
   Eine id für die Road- bzw. Path-Elemente kann dem *Road Header Record* entnommen werden.

4. *Länge*
   Als Länge für die Road bzw. Path-Elemente kann die im *Road Header Record* hinterlegte Längenangabe verwendet werden.

5. *Name*
   Ein Name für die Road- bzw. Path-Elemente kann dem *Road Header Record* entnommen werden.

Hier ist noch anzumerken, dass ein Path-Eintrag laut OpenDRIVE®-Spezifikation entweder nur linke oder nur rechte Lane-Einträge besitzt. Folglich sollte ein Path-Element auch entweder nur linke oder nur rechte Connector-Elemente besitzen.

Informationen für Junction-Elemente

Junction-Elemente benötigen zur Simulation eine Liste mit zugehörigen Connector-Elementen sowie eine Liste welche die Vorfahrtsregelungen/-prioritäten zwischen diesen Connector-Elementen beschreibt. Zusätzlich sind an Junction-Elementen Informationen wie id und Name sowie weitere Informationen zum repräsentierten Verzweigungsabschnitts möglich.

1. *Liste mit Connector-Elementen*

	Primär
1.	Liste mit Connector-Elementen
2.	Vorfahrtsprioritäten
	Optional
3.	id
4.	Name
5.	Liste mit Informationen

Tabelle 10: Informationen für Junction-Elemente.

	Primär
1.	Position (X- und Y-Koordinate)
	Optional
2.	Liste mit Informationen

Tabelle 11: Informationen für Waypoint-Elemente.

Jedes Junction-Element benötigt eine Liste mit zugehörigen Connector-Elementen. Diese Liste kann aus Informationen in den *Connection Record* Einträgen eines *Junction Record*-Eintrags erzeugt werden. Dazu müssen alle Connector-Elemente die deren id in einem *connectingRoad* Attribut referenziert ist berücksichtigt werden.

2. *Vorfahrtsprioritäten*
   Es wird zusätzlich eine Liste mit Prioritäten benötigt, welche die Vorfahrtsregelungen zwischen den Connector-Elementen beschreibt. Um solche Prioritäten zu gewinnen bietet es sich an Informationen aus dem *Junction Priority Record* auszuwerten. An dieser Stelle sind jeweils Referenzen auf zwei Road-Einträge hinterlegt, dabei ist ein Eintrag als “high” also höher priorisiert und der andere als “low” also niedriger priorisiert gekennzeichnet.

3. *id*
   Optional kann jedem Junction-Element eine id zugewiesen werden. Eine entsprechende Information ist als Parameter im *Junction Record* vorhanden.

4. *Name*
   Wie bei der id-Information kann jedem Junction-Element ein Name zugewiesen werden. Eine entsprechende Information ist als Parameter im *Junction Record* vorhanden.

5. *Liste mit Informationen*
   Jedes Junction-Element kann zudem eine Liste mit zusätzlichen Informationen besitzen. Welche Informationen an dieser Stelle integriert werden könnten und welche sinnvollen Daten die OpenDRIVE®-Spezifikation dafür bietet, wird in Abschnitt 3.2.3 betrachtet.

Informationen für Waypoint-Elemente

Damit eine Navigation stattfinden kann benötigt ein Waypoint-Element in erster Linie eine Position in der Ebene. Zusätzlich ist die Hinterlegung weiterer Informationen möglich. Welche Informationen an dieser Stelle sinnvoll sind und was der OpenDRIVE®-Standard an Informationen bietet wird in Abschnitt 3.2.3 betrachtet.

1. *Position (X- und Y-Koordinate)*

Wie mögliche Positionen für Waypoint-Elemente berechnet werden können wurde bereits im Abschnitt *Informationen für Lane-Elemente* unter dem Punkt *Liste mit Waypoint-Elementen* betrachtet.

2. *Liste mit Informationen*
   Jedes Waypoint-Element kann zudem eine Liste mit zusätzlichen Informationen besitzen. Welche Informationen an dieser Stelle integriert werden könnten und welche sinnvollen Daten die OpenDRIVE®-Spezifikation dafür bietet, wird in Abschnitt 3.2.3 betrachtet.

3.2.3. Zusätzliche Informationen für Junction- und Waypoint-Elemente

Nachdem im vorherigen Abschnitt die für die die mikroskopische Ebene notwendigen Parameter diskutiert wurden soll in diesem Abschnitt betrachtet werden, welche zusätzlichen Informationen innerhalb der Waypoint- und Junction-Elemente hinterlegt werden können und welche Informationen die OpenDRIVE®-Spezifikation dafür bietet.

Waypoint-Elemente

Erste einfache Beispiele für solche zusätzlichen Informationen sind in der OpenDRIVE®-Spezifikation definiert. Hier bieten sich die Informationen der *Road Description Records* (siehe 2.2.2) an. Zu jedem Lane-Eintrag können jeweils mehrere Einträge jedes *Road Description Records* bestehen. So können Informationen zu Fahrbahnmarkierungen, Fahrbahnmaterial, Sichtverhältnissen im Umfeld einer Spur, Geschwindigkeitsbegrenzungen, Zugangsbeschränkungen und Spurhöhen gewonnen werden. Zu jeder dieser Informationen können im Verlauf eines einzelnen Lane-Eintrags theoretisch unendlich viele Einträge bestehen.

Die nächste Information, welche in einem Waypoint-Element hinterlegt werden kann, sind Informationen über benachbarte Lane-Elemente also über Nachbarspuren. Es genügt nicht diese Information innerhalb eines Lane-Elements zu hinterlegen, da sich die Nachbarspuren innerhalb des Elements ändern können. In OpenDRIVE®ist die Anzahl der Lane-Einträge innerhalb eines LaneSection-Eintrags konstant, folglich bleiben auch die Nachbarspuren die selben. Es reicht also aus während der Generierung bei jedem Wechsel des LaneSection-Eintrags die Nachbarspuren in einem entsprechenden Waypont-Element zu hinterlegen. Die Information bleibt bis zum nächsten LaneSection-Eintrag gültig. Gesondert könnte hier der Beginn von Abbiege- oder Überholspuren vermerkt werden um später nicht mehr zwischen normalen Fahrspuren und diesen unterscheiden zu müssen. Dies würde das Hervorrufen der Entscheidung eines Spurwechselmanövers an solchen Punkten vereinfachen bzw. von Spurwechseln aus Grund von Überhol- oder Ausweichmanövern einfacher unterscheidbar machen.

Es könnten auch Informationen zu Verkehrszeichen entlang der Straße in den Waypoint-Elementen hinterlegt werden. Dies können statische Signale (Schilder, auf die Straße gemalte Geschwindigkeitsbegrenzungen, ...) oder dynamische Signale (Ampeln, ...) sein. Informationen zu Signalen entlang der Straße sind im *Road Signals Record* zu finden.

Zusätzlich ist es möglich über Waypoint-Elemente Informationen zu Orten zu hinterlegen. Ein Ort kann verschiedenstes sein: Die erste Möglichkeit wäre es bestimmte Gegebenheiten wie industrielle- (Firmengebäude mit Arbeitsplätzen, etc.), gewerbliche- (Geschäfte, Dienstleister, etc.), öffentliche Einrichtungen (Schulen, Krankenhäuser, etc.) oder Wohnhäuser anzugeben. Später könnte man über diese Informationen Agenten aktiv zu diesen Orten navigieren lassen. Eine weitere Möglichkeit wäre es Parkplätze zu hinterlegen. Ein Agent könnte bei erreichen eines solchen Wegpunkts feststellen, ob der Parkplatz belegt ist und wenn nicht, den Platz einnehmen. Leider bietet die OpenDRIVE®-Spezifikation keine Informationen in dieser Richtung an, da sie hauptsächlich für die Beschreibung der Straßennetzwerke an sich geschaffen wurde.

Junction-Elemente

Als Zusatzinformation könnten an Junction-Elementen z.B. Informationen über, den repräsentierten Kreuzungsbereich betreffende Signale hinterlegt werden. Hier kann wieder zwischen statischen und dynamischen Signalen unterschieden werden. Informationen zu den Signalen können aus den *Road Signal Records* gewonnen werden. Zusätzlich ist es möglich aus dem *Junction Controller Record* und dem *Controller Record* Informationen zur Koordination von dynamischen Signalen zu gewinnen.

Ergänzende Informationen (Ancillary Data)

Ein wichtiger Aspekt innerhalb der OpenDRIVE®-Spezifikation ist die definierte Eintragsform *userData*. Diese ermöglicht es, eigene Informationen an jeder Stelle innerhalb von OpenDRIVE®-Dateien zu hinterlegen und so den Standard für eigene Zwecke zu erweitern.

4. SeRoNet: Generierung semantischer Verkehrsnetzwerke

Dieses Kapitel befasst sich mit der Realisierung der Inhalte aus Kapitel 3. Um diese zu beschreiben werden beginnend einige während des Projektes verwendete Tools, darunter die Unity Game Engine und der Trian3D Builder, eingeführt und einige wichtige Aspekte näher erläutert. Anschließend werden nach und nach einzelne Teile der Implementierung beschrieben. Begonnen wird mit der Beschreibung einer objektorientierten Darstellung der OpenDRIVE®-Daten. Anschließend wird das Verkehrsnetzwerkmodell für die mesoskopische Simulationsebene beschrieben, bevor auf den dafür implementierten Generierungsprozess eingegangen wird. Danach folgt die Beschreibung der für diese Arbeit implementierten Netzwerkelemente für die mikroskopische Simulationsebene, bevor letztendlich der Generierungsprozess beschrieben wird, der auf Basis dieser Elemente Verkehrsnetzwerke für die mikroskopische Simulationsebene zusammensetzt.

4.1. Tools

In diesem Abschnitt werden einige im Rahmen der Realisierung verwendete Werkzeuge eingeführt und auf einzelne für die Realisierung wichtige Aspekte näher eingegangen. Begonnen wird mit der Unity Game Engine, welche als Simulationsumgebung genutzt wird. Anschließend wird die Trian3D Builder Software eingeführt. Diese wird zur Generierung von OpenDRIVE®-Datensätzen verwendet, welche für Tests während des Entwicklungsprozesses, sowie zur Evaluation benötigt werden. Letztlich wird auf zwei Tools eingegangen, welche im Vorhinein zum Projekt entwickelt wurden. Erstens eine Bibliothek (xodrReader), welche es ermöglicht OpenDRIVE®-Daten in die Simulationsumgebung zum importieren und dadurch nutzbar zu machen und zweitens ein externes Programm (xodrViewer) mit dem durch die Bibliothek OpenDRIVE®-Daten eingelesen und listenartig dargestellt werden können.

4.1.1. Unity Game Engine

Momentan wird in den Projekten FIVIS und AVeSi die Unity Game Engine als Simulationsumgebung genutzt. In dieser ist es möglich, Skripte verschiedener Programmiersprachen[12] zu erstellen. Es besteht eine umfangreiche API zum Zugriff auf die einzelnen Funktionalitäten der Engine. Die Repräsentation von Elementen in der Engine geschieht über so genannte GameObjects. An jedes GameObject können Skripte der erwähnten Programmiersprachen angehängt werden. In diesen Skripten werden Funktionen zur Verfügung gestellt, welche abgeleitet werden können und bei bestimmten Ereignissen aufgerufen werden.

GameObjects und Scripting

An dieser Stelle soll etwas genauer auf die, für die Umsetzung wichtige, Skriptumgebung innerhalb von Unity eingegangen werden.

> Scripting inside Unity consists of attaching custom script objects called behaviours to game objects. Different functions inside the script objects are called on certain events. [22]

Alle Dinge / Objekte / Entitäten werden innerhalb der Unity Engine durch so genannte GameObjects repräsentiert. Das Skripten funktioniert auf die Weise, dass an die verschiedenen GameObjects jeweils Skriptobjekte anhängt werden können. Diese Skriptobjekte sind Codestücke, die von der *MonoBehaviour* Klasse abgeleitet sind. MonoBehaviour stellt verschiedene Funktionen zur Verfügung durch deren ableiten unterschiedlichste Funktionalitäten und Verhaltensweisen an die GameObjects angefügt werden können. Einige wichtige Funktionen sind folgend beschrieben.

[12] JavaScript, C# und Boo.

Funktionsname	Funktionsbeschreibung
Update	Wird in jedem Frame aufgerufen unter der Bedingung, dass das Objekt aktiv ist.
LateUpdate	Wird in jedem Frame aufgerufen unter der Bedingung, dass das Objekt aktiv ist.
FixedUpdate	Wenn das Objekt aktiv ist wird die Funktion in festen Frameintervallen aufgerufen.
Awake	Wird beim Laden der Skriptinstanz aufgerufen.
Start	Wird aufgerufen bevor irgendeine der Update-Methoden erstmals ausgeführt wird.
OnDrawGizmos	Wird in jedem Frame aufgerufen.
OnDrawGizmosSelected	Wird zusätzlich in jedem Frame aufgerufen unter der Bedingung, dass das zugeordnete GameObject aktiv, also angewählt ist.

Tabelle 12: Durch MonoBehaviour bereitgestellte Funktionen. Die aufgelisteten Funktionen werden durch die MonoBehaviour Klasse zur Verfügung gestellt. Sie werden beim Auftreten der beschriebenen Ereignisse aufgerufen.

Die beiden Funktionen OnDrawGizmos und OnDrawGizmosSelected können als Debughilfe verwendet werden. Mit ihnen ist es möglich, GameObjects auf eine einfache Weise zu visualisieren. Es stehen verschiedene vorgefertigte Gizmo-Funktionen zur Verfügung z.B. Ray, Line, Cube und Sphere.

4.1.2. Trian3D Builder

Der von der TrianGraphics GmbH[13] entwickelte Trian3D Builder ist eine Software zur Generierung virtueller Landschaften. Es existiert ein optionales Roads Modul welches die Erstellung von Verkehrsnetzwerken ermöglicht. Durch eine Exportfunktion ist es möglich, eine 3D-Szene, welche in der Unity Game Engine verwendet werden kann zu erzeugen. Zusätzlich kann ein passender OpenDRIVE®-Datensatz erzeugt werden, dazu wird jedoch das Roads Modul Plus benötigt. Ein weiteres Feature des Tools ist es Vektordaten, welche z.B. aus OpenStreetMap-Daten[14] erzeugt werden können, zu importieren. Idealerweise können so im späteren Projektverlauf verschiedene, auf realen Daten basierende OpenDRIVE®-Datensätze erzeugt werden.

4.1.3. xodrReader und xodrViewer

xodrReader ist eine in C# implementierte Softwarebibliothek, welche im Vorfeld des Projekts entwickelt wurde. Durch die Bibliotheksfunktionen ist es möglich, die in einer OpenDRIVE®-Datei hinterlegten Daten in eine dem XML-Aufbau ähnliche Datenstruktur einzulesen. Es ist möglich, die Bibliotheksdatei in ein Unity Projekt einzubinden. So kann innerhalb der Simulationsumgebung auf die in einer OpenDRIVE®-Datei hinterlegten Informationen zugegriffen werden. Zusätzlich besteht eine eigenständige Applikation, mit welcher eingelesene OpenDRIVE®-Daten dargestellt werden können, der xodrViewer. Mit diesem können unter Verwendung der xodrReader-Bibliothek OpenDRIVE®-Dateien eingelesen und listenartig dargestellt werden. Dies kann zu Debugzwecken genutzt werden, so muss nicht immer die OpenDRIVE®-Datei selber betrachtet werden.

[13] www.triangraphics.de
[14] www.openstreetmap.de

4.2. Objektorientierte Darstellung der OpenDRIVE®-Daten

Um die Nutzung der im OpenDRIVE®-Format beschriebenen Daten innerhalb der Simulationsumgebung zu erleichtern, wurde eine objektorientierte Darstellung der OpenDRIVE®-Daten konzipiert und implementiert. Darauf aufbauend wurde eine Überführung der durch die xodrReader-Bibliothek eingelesenen Daten in diese Datenstruktur realisiert. Im Anhang ist eine UML-ähnliche Darstellung der objektorientierten Datenstruktur vorhanden.

4.3. Verkehrsnetzwerkmodell für warteschlangenbasierte Mesosimulation

Die Agenten außerhalb des Wirkungsbereich des Benutzers sollen über ein warteschlangenbasiertes Mesosimulationsmodell berechnet werden. Dazu muss ein Graph erzeugt werden, der gewisse Daten, wie Anzahl der Spuren einer Straße, maximale Geschwindigkeit, etc. zur Verfügung stellt. Die von den Elementen des Graphen benötigten Informationen wurden bereits im vorherigen Kapitel betrachtet. Folgend werden die Grundelemente der Graphen noch einmal beschrieben. Weitere Informationen können der Masterarbeit von T. Dettmar entnommen werden [5]. Im Anschluss daran wird erläutert, wie die Überführung von OpenDRIVE®-Daten in einen entsprechenden Simulationsgraphen erfolgt.

4.3.1. Elemente des Modells für die mesoskopische Simulationsebene

Jeder mesoskopische Verkehrsnetzwerksgraph besteht aus drei Grundelementen (Node-Element, Edge-Element und Wormhole-Element) die folgend erläutert werden.

Node-Element

Node-Elemente stellen die Kreuzungen bzw. Verzweigungsabschnitte innerhalb des Verkehrsnetzwerksgraph dar, hier werden Tabellen mit eingehenden und ausgehenden Kanten verwaltet. Über eine Tabelle im Node-Element werden Wahrscheinlichkeiten verwaltet über die entschieden wird, welche nachfolgenden Edge-Elemente für ankommende Agenten ausgewählt werden.

Edge-Element

Edge-Elemente repräsentieren die Straßen im Graphen. Sie besitzen jeweils eine Referenz auf ein vorangehendes und ein nachfolgendes Node- bzw. Wormhole-Element sind also gerichtet, was bedeutet, dass für Spuren jeder Fahrtrichtung ein eigenständiges Edge-Element erzeugt werden muss. Einem Edge-Element werden Informationen wie die Länge des repräsentierten Straßenabschnittes, die Anzahl der Spuren und die durchschnittliche Maximalgeschwindigkeit des repräsentierten Straßenabschnitts zugeordnet.

Wormhole-Element

Das Wormhole-Element wurde im Rahmen dieser Arbeit zusätzlich eingeführt, um den Fall abzudecken, dass eine Straße in der Beschreibung entweder keinen Vorgänger oder keinen Nachfolger besitzt. Wird ein Agent in ein Wormhole-Element geleitet, wird er vom Edge-Element entfernt und in ein zufälliges anderes Wormhole-Element eingefügt, von welchem aus er wieder das Verkehrsnetzwerk betritt. Über diese Vorgehensweise ist gewährleistet, dass keine Agenten das Netzwerk verlassen und somit eine konstante Anzahl von Agenten im Netzwerk erhalten bleibt.

4.3.2. Generierungsprozess

Für jedes Element der Verkehrsnetzwerksgraphen wird innerhalb der Simulationsumgebung ein GameObject erzeugt und anschließend ein entsprechendes C# Skript (Node-/Edge-/Wormhole-Script) daran angehängt. Ein GameObject, an das ein entsprechendes Skript angehängt wurde, wird folgend mit *-Element bezeichnet, wobei an Stelle des Sterns der Typ des Elements steht. Die Oberklasse der Skripte (*Monobehaviour*) stellt verschiedene Methoden zur Verfügung, welche kontinuierlich durch die Engine aufgerufen werden: Einerseits Methoden, die für die Visualisierung des GameObjects verantwortlich sind und andererseits Methoden, in denen in diesem Fall die für die Simulation notwendigen Berechnungen stattfinden. Siehe dazu auch Abschnitt 4.1.1. Die zur Berechnung benötigten Parameter müssen jedem Skript bei der Erzeugung übergeben werden. Welche Parameter benötigt werden und woher diese Informationen entnommen werden können, wurde im vorherigen Kapitel bereits beschrieben.

Road-Einträge → Edge-Elemente

Aus jedem Road-Eintrag in der Verkehrsnetzwerksbeschreibung, welcher keinen Pfad darstellt, werden zwei Edge-Elemente generiert. Jedes der beiden Edge-Elemente repräsentiert alle Spuren der Straße, die in eine gemeinsame Richtung verlaufen. Die Parameter für ID und Länge der Straße können direkt aus dem Road-Eintrag übernommen werden. Start- und End-Node-Element können aus den im Road-Element befindlichen Predecessor- und Successor-Einträgen ermittelt werden. Die Anzahl der Spuren wird aus den für das Fahren relevanten Lane-Einträgen ermittelt (Lane-Einträge für Bürgersteige, Seitenstreifen, etc. werden an dieser Stelle also nicht berücksichtigt). Lane-Einträge mit negativer ID zählen zum einen Edge-Element und Lane-Einträge mit positiver ID zur anderen. Die maximale Geschwindigkeit wird über die in den einzelnen Teilabschnitten des Road-Eintrags gültigen Geschwindigkeiten und unter Berücksichtigung derer Längen gemittelt.

Die zur Simulation benötigten Daten wurden somit vollständig betrachtet. Um eine Visualisierung des Graphen erstellen zu können, werden jedoch zusätzlich Positionsdaten benötigt. Die Startposition eines Edge-Elements wird direkt aus der OpenDRIVE®-Beschreibung ermittelt und als Position des zu Grunde liegenden GameObjects verwendet. Der Verlauf des Edge-Elements wird durch eine Gerade, die zwischen den Positionen des referenzierten Start- und End-Node-Elements gezogen wird, dargestellt. Der echte Verlauf der Straße ist an dieser Stelle nicht von Bedeutung.

Junction-Einträge → Node-Elemente

In Node-Elementen werden folgende Informationen zur Simulation benötigt: Eine Liste mit nachfolgenden Edge-Elementen, eine Liste mit vorangehenden Edge-Elementen sowie eine Matrix mit Übergangswahrscheinlichkeiten für die entsprechenden nachfolgenden Edge-Elemente.

Die zur Erzeugung der Node-Elemente benötigten Informationen werden aus den Junction-Einträgen der Straßennetzwerksbeschreibung abgeleitet. Zur Simulation wird in den Node-Elementen nur jeweils eine Liste der eingehenden und eine Liste der ausgehenden Edge-Elemente benötigt. Diese Informationen werden aus den Predecessor- sowie Successor-Einträgen der Road-Einträge ermittelt.

Die zur Visualisierung benötigte Position der Node-Elemente wird durch einen Mittelwert aus den Startpunkten der im Junction-Eintrag referenzierten Path-Einträge bestimmt. Dies geschieht, weil Junction-Einträge von sich aus keine Koordinaten besitzen.

Wormhole-Elemente

Es gibt zwei Fälle, in denen es nötig ist, ein Wormhole-Element zu erzeugen. Nämlich wenn ein Road-Eintrag in der Beschreibung keinen Prececessor- oder keinen Successor-Eintrag besitzt. Für das Wormhole-Element benötigte Informationen sind Referenzen auf ein in das Wormhole-Element hineinführendes Edge-Element sowie eine Liste mit jedem aus einem bestehenden Wormhole-Element ausgehenden Edge-Element. Als eingehendes Edge-Element wird jeweils das zum Wormhole-Element hinführende Edge-Element verwendet und als ausgehendes Edge-Element wird jeweils das vom Wormhole-Element wegführende Edge-Element verwendet. An dieser Stelle soll angemerkt sein, dass es auch Wormhole-Elemente mit nur eingehendem oder nur ausgehendem Edge-Element geben kann, nämlich wenn in die entsprechende Richtung keine relevanten Spuren verlaufen. Wenn alle Wormhole-Elemente erzeugt wurden, werden diesen anschließend die Informationen über die ausgehenden Edge-Elemente der anderen Wormhole-Elemente zugewiesen. Wie bereits beschrieben werden Wormhole-Elemente an den Stellen erzeugt, wo eine Straße entweder keinen Vorgänger oder keinen Nachfolger besitzt. Die Position eines Wormhole-Elements wird also entweder am Start- oder am Endpunkt einer Straße befindlich sein. Der Straßenstartpunkt ist bereits innerhalb der Road-Einträge beschrieben und wird direkt verwendet. Der Endpunkt einer Straße muss über die Geometriedefinition im Road-Eintrag berechnet werden.

4.4. Verkehrsnetzwerkmodell für kognitive Mikrosimulation

Im vorangehenden Abschnitt wurde bereits der umgesetzte Generierungsprozess für das Verkehrsnetzwerkmodell für warteschlangenbasierte Mesosimulation beschrieben. In diesem Abschnitt wird nun die Realisierung des Verkehrsnetzwerkmodells für kognitive Mikrosimulation betrachtet. Dazu werden zuerst die Elemente des Modells und deren Visualisierungen erläutert, bevor anschließend der realisierte Prozess erklärt wird, durch den es möglich ist, automatisiert Verkehrsnetzwerke, die auf diesen Elementen basieren, zu erzeugen.

4.4.1. Elemente des Modells für die mikroskopische Simulationsebene

In diesem Abschnitt werden die realisierten Elemente für das mikroskopische Verkehrsnetzwerkmodell beschrieben. Zu jedem Element existiert eine Beschreibung sowie eine Abbildung der Visualisierung (Abbildungen 12 - 16).

Lane-Element

Ein Lane-Element besteht aus einer Menge von Waypoint-Elementen sowie einer Menge von vorangehenden Connector-Elementen und einer Menge von nachfolgenden Connector-Elementen. In Abbildung 12 ist die Visualisierung eines Lane-Elements dargestellt. Das Lane-Element selbst wird durch die gelbe Linie dargestellt, welche entlang der zugehörigen Waypoint-Elemente verläuft. Die Waypoint-Elemente werden durch die zu sehenden Würfel repräsentiert. Weiterhin können links im Bild zwei vorangehende Connector-Elemente (grün) und rechts im Bild zwei nachfolgende Connector-Elemente (rot) betrachtet werden.

Connector-Element

Ein Connector-Element besteht aus einer Menge von Waypoint-Elementen sowie einem vorangehenden und einem nachfolgenden Lane-Element. In Abbildung 13 ist die Visualisierung eines Connector-Elements dargestellt. Das Connector-Element selbst wird durch die gelbe Linie dargestellt, welche entlang der zugehörigen Waypoint-Elemente verläuft. Die Waypoint-Elemente werden durch die zu sehenden Würfel repräsentiert. Weiterhin kann links im Bild das nachfolgende Lane-Element (rot) und rechts im Bild das vorangehende Lane-Element (grün) betrachtet werden.

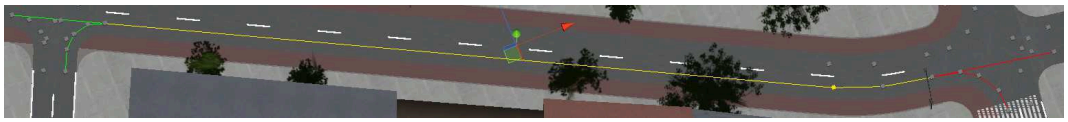


Abbildung 12: Visualisierung eines Lane-Elements. Die gelbe Linie stellt das Lane-Element dar. Die Würfel entlang der gelben Linie sind die Waypoint-Elemente des Lane-Elements. Links und rechts sind jeweils zwei vorangehende Connector-Elemente (grün) und zwei nachfolgende Connector-Elemente (rot) zu sehen.

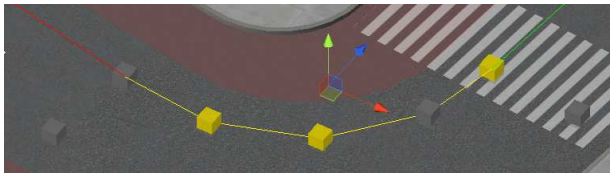


Abbildung 13: Visualisierung eines Connector-Elements. Die gelbe Linie stellt das Connector-Element dar. Die Würfel entlang der gelben sind die Waypoint-Elemente des Connector-Elements. Links und rechts ist das vorangehende (grün) sowie das nachfolgende Lane-Element (rot) zu sehen.

Waypoint-Element

Ein Waypoint-Element ist Teil eines Lane- oder Connector-Elements und dient dazu dessen Geometrie zu beschreiben. Dazu besitzt es eine X- und eine Y-Koordinate sowie weitere für diese Position oder ab dieser Position relevante Informationen. In Abbildung 14 ist die Visualisierung mehrerer Waypoint-Elemente innerhalb eines Connector-Elements dargestellt. Das angewählte Waypoint-Element wird als gelber Würfel dargestellt. Die grauen Würfel sind andere Waypoint-Elemente des selben Connector-Elements.

Road- und Path-Element

Road- bzw. Path Elemente vereinigen jeweils eine Menge nebeneinander verlaufender Lane- bzw. Connector-Elemente. In Abbildung 15 ist die Visualisierung eines Road-Elements dargestellt. Path-Elemente mit mehreren Connector-Elementen sind relativ selten, da dies bedeuten würde das mehrere Abbiegespuren in eine gemeinsame Richtung existieren. Da die Visualisierung der beiden Elemententypen jedoch identisch ist kann hier stellvertretend ein Road-Element betrachtet werden. In der Abbildung ist ein Road-Element mit zwei Lane-Elementen dargestellt. Die orange- und gelbfarbenen Linien stellen bei der Visualisierung die zugehörigen Lane-Elemente dar. Wobei alle Lane-Elemente die in eine Richtung verlaufen orange sind und alle Lane-Elemente die in die andere Richtung verlaufen gelb sind. Nachfolger- und Vorgängerelemente werden bei dieser Visualisierung nicht betrachtet.

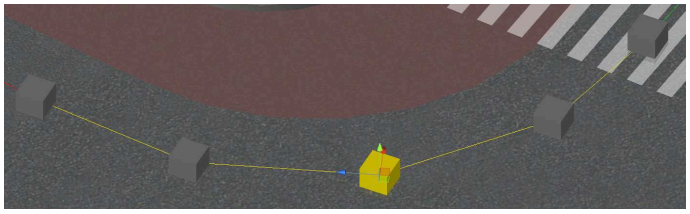


Abbildung 14: Visualisierung eines Waypoint-Elements. Der gelbe Würfel stellt ein Waypoint-Element innerhalb eines Lane- oder Connector-Elements (gelbe Linie) dar.

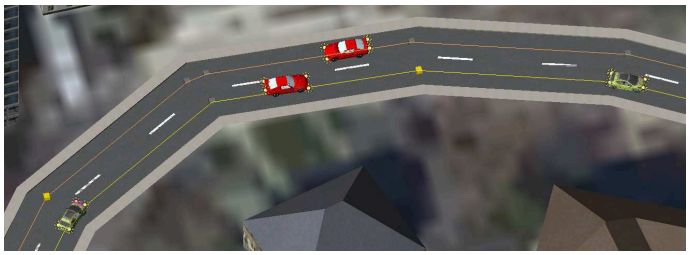


Abbildung 15: Visualisierung eines Road-Elements. Die in eine Richtung verlaufenden Lane-Elemente werden als orangefarbene Linie dargestellt. Die Lane-Elemente die in die entgegengesetzte Richtung verlaufen sind als gelbe Linie dargestellt.

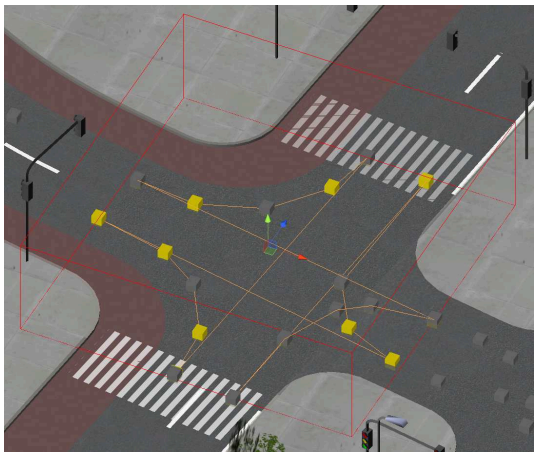


Abbildung 16: Visualisierung eines Junction-Elements. Der rote Gitterwürfel stellt das Junction-Element dar. Innerhalb des Würfels befinden sich die zugehörigen Connector-Elemente (orange) und ihre Waypoint-Elemente.

Junction-Element

Ein Junction-Element vereint eine Menge von Connector-Elementen welche zu einem gemeinsamen Kreuzungsabschnitt gehören. Im Junction-Element können Informationen zu Vorfahrtsregelungen zwischen den Connector-Elementen hinterlegt werden. In Abbildung 16 ist die Visualisierung eines Junction-Elements dargestellt. Der rote Gitterwürfel repräsentiert das Junction-Element, die orangefarbenen Linien im Würfel repräsentieren die zugehörigen Connector-Elemente und die gelben und grauen Würfel die Waypoint-Elemente zu den Connector-Elementen.

4.4.2. Generierungsprozess

Der Erzeugungsprozess, welcher aus den gegebenen OpenDRIVE®-Daten das Verkehrsnetzwerk für die Mikrosimulationsebene generiert, lässt sich in mehrere Teilschritte untergliedern. Die Abbildung 17 zeigt zwei Pseudocodeabschnitte, welche die Teilschritte des Erzeugungsprozess darstellen. Als erstes werden Road- und Path-Elemente aus den in der OpenDRIVE®-Beschreibung zu findenden Road-Einträgen erstellt. Hierbei werden aus Road-Einträgen in denen ein Junction-Eintrag referenziert ist Path-Elemente erstellt und aus Road-Einträgen ohne referenzierten Junction-Eintrag Road-Elemente. Anschließend werden Lane- und Connector-Elemente als Kindelemente der entsprechenden Road- bzw. Path-Elemente erstellt. Nach der Erzeugung der Lane- und Connector-Elemente wird anhand der OpenDRIVE®-Geometriebeschreibung eine Menge von Positionen für potentielle Waypoint-Elemente berechnet. Die Menge dieser berechneten Positionen wird anschließend reduziert. An den verbliebenen Positionen werden Waypoint-Elemente erzeugt. Nachdem alle Lane- und Connector-Elemente erzeugt und mit Waypoint-Elementen ausgestattet wurden, werden Verbindungen zwischen den Elementen gesetzt. Dazu werden alle Road-Einträge mit referenziertem Junction-Eintrag – also Path-Einträge – durchlaufen und Verbindun-

```
1  WayPointSystem()
2  {
3      //Fuer jedes Road-Element in der Strassenbeschreibung ein Segment erstellen
4      GenerateSegments();
5      //Verbindungen zwichen Lane- und Connector-Elementen setzen
6      SetConnections();
7      //Zu bestehenden Waypoint-Elementen zusaetzliche Informationen hinzufuegen
8      TagWaypoints();
9      //Waypoint-Elemente mit zusaetzlichen Informationen erzeugen
10     GenerateAddintionalWaypoints();
11     //Junction-Elemente erzeugen
12     GenerateJunctions();
13 }
14
15 void GenerateSegments()
16 {
17     foreach r in roads
18     {
19         Segment s = CreateSegment();
20
21         //Fuer das Segment in gegebenem Abstand Wegpunkte erstellen
22         GenerateWaypoints(s, DISTANCE);
23         //Folge an erzeugten Wegpunkten reduzieren (EPSILON = maximale Abweichung)
24         s.wps = DouglasPeucker(s.wps,EPSILON);
25
26         segmentList.add(s);
27     }
28 }
```

Abbildung 17: Pseudocode zum Generierungsprozess für Verkehrsnetzwerke der Mikrosimulationsebene. In diesem Pseudocode sind die einzelnen Schritte für die Generierung von Verkehrsnetzwerken der Mikrosimulationsebene dargestellt.

gen entsprechend derer Predecessor- und Successor-Einträge gesetzt. Dabei wird die Verbindung von Connector-Element zu Lane-Element und gleichzeitig auch die Verbindung von Lane-Element zu Connector-Element gesetzt. Da jedes Connector-Element eindeutig mit einem vorangehenden und einem nachfolgenden Lane-Element verbunden ist, ist gewährleistet, dass auch alle Lane-Connector-Verbindungen – welche nicht eindeutig sind – berücksichtigt wurden. Im dritten und vierten Generierungsschritt werden zuerst zusätzliche Informationen an bestehende Waypoint-Elemente angefügt und anschließend zusätzliche Waypoint-Elemente mit zusätzlichen Informationen eingefügt. Letztendlich werden im fünften Schritt Junction-Elemente erstellt und an die entsprechenden Connector-Elemente angehängt. Die einzelnen Schritte werden folgend näher beschrieben.

Schritt 1: Road- und Path-Elemente erzeugen

Als erstes werden Road- und Path-Elemente erzeugt. Dazu werden die Road-Einträge der OpenDRIVE®-Datei durchlaufen. Für jeden Road-Eintrag dessen Attribut *junction* den Wert -1 hat wird ein Road-Element erzeugt und für jeden anderen Road-Eintrag, also für die Einträge mit einer Referenz auf einen Junction-Eintrag, ein Path-Element. Nachdem ein Element erzeugt wurde müssen mehrere Unterschritte ausgeführt werden. Für das Element muss eine Anzahl an entsprechenden Lane- bzw. Connector-Elementen erzeugt werden, für welche wiederum eine Anzahl an Waypoint-Elementen erzeugt werden muss.

Lane- und Connector-Elemente erzeugen

Um die Lane- bzw. Connector-Elemente zu erzeugen müssen die zum jeweiligen Road-Eintrag gehörenden Lane-Einträge durchlaufen werden. Hier muss beachtet werden, dass diese sich über mehrere LaneSection-Einträge erstrecken können. Zur Generierung wird das Road-Element über die geometrische Länge iteriert – genauer gesagt sogar zweimal iteriert einmal von vorne nach hinten also von 0 bis zur Länge der Straße um Lane-Elemente bzw. Connector-Elemente für die in Beschreibungsrichtung rechten Spuren zu erzeugen und einmal von hinten nach vorne also von der

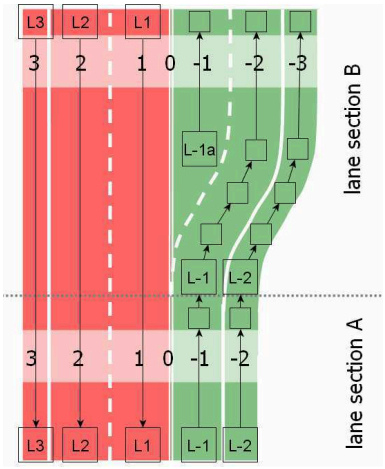


Abbildung 18: Vereinigung von Lane-Einträgen verschiedener LaneSection-Einträge. Abbildung nach [7, S.14].

Länge der Straße bis 0 um Lane-Elemente bzw. Connector-Elemente für die in Beschreibungsrichtung linken Spuren zu erzeugen. Zu jedem Lane-Eintrag auf dem ersten LaneSection-Eintrag wird ein entsprechendes Lane- bzw. Connector-Element erstellt. Während des Iterierens wird zu jeder Länge der gültige LaneSection-Eintrag ermittelt. Wenn sich dieser ändert muss beachtet werden, dass nun auch eine neue Menge an Lane-Einträgen gültig ist. An dieser Stelle müssen zwei Fälle unterschieden werden:

1. Im ersten Fall besteht ein Lane-Eintrag der über einen Link-Eintrag mit einem Lane-Eintrag in der vorherigen LaneSection verbunden ist. In diesem Fall muss das bestehende Lane-Element bzw. Connector-Element über diesen Lane-Eintrag erweitert werden. Es findet also eine Vereinigung der Lane-Einträge zu einem einzigen Lane- bzw. Connector-Element statt.

2. Der zweite Fall besteht wenn ein Lane-Eintrag keinen Vorgänger besitzt, also ein neuer Abschnitt beginnt. In diesem Fall muss ein neues Lane- bzw. Connector-Element erzeugt werden.

Hier sollte auch erwähnt werden, dass es möglich ist, dass Lane-Einträge des vorherigen LaneSection-Eintrags enden. Die zugehörigen Lane-Elemente können als abgeschlossen markiert werden. In Abbildung 18 ist ein Beispiel für den Fall, dass mehrere LaneSection-Einträge bestehen dargestellt. Man kann sehen, dass Lane -1 aus LaneSection A mit Lane -2 auf LaneSection B vereinigt werden muss. Für die auf LaneSection B neu beginnende Lane -1 muss ein neues Lane-Element erzeugt werden.

Waypoint-Elemente erzeugen

Während des Iterierens über die Länge der Road-Elemente werden zusätzlich die Waypoint-Elemente für die Lane- und Connector-Elemente erstellt. Die Erzeugung der Waypoint-Elemente untergliedert sich nochmals in zwei Teilschritte. In jedem Längenschritt werden für alle Lane-Elemente mögliche Positionen für Waypoint-Elemente berechnet. Somit besteht nach dem iterieren für jedes Lane-Element eine mögliche Folge von potentiellen Positionen für Waypoint-Elemente. Die Positionen befinden sich in festen Distanzen links- bzw. rechts entlang der Straßenreferenzlinie. Die Positionsfolgen für jedes Lane- bzw. Connector-Element werden anschließend reduziert, wobei gewährleistet wird, dass die Form des Kantenzugs innerhalb einer gegebenen Toleranz erhalten bleibt. Anschließend werden Waypoint-Elemente an den übrig gebliebenen Positionen der Folge erzeugt.

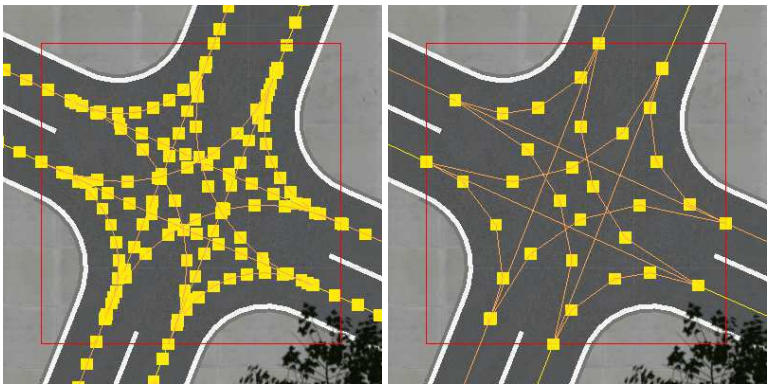


Abbildung 19: (a) Kreuzungsabschnitt mit Waypoint-Elementen in Abständen von einem Meter. ($\sim$ 140 Waypoint-Elemente) (b) Kreuzungsabschnitt mit einer durch den Douglas-Peucker-Algorithmus reduzierten Menge an Waypoint-Elementen. (48 Waypoint-Elemente bei $\epsilon = 0,3m$)

Folge von Positionen für mögliche Waypoint-Elemente berechnen

Die Positionen werden wie bereits in Abschnitt 3.2.2 (Formel 6 und 7) beschrieben berechnet. Nach der Berechnung besteht für jedes Lane- bzw. Connector-Element eine Menge an möglichen Positionen für Waypoint-Elemente.

Folge an möglichen Positionen minimieren (Douglas-Peucker Algorithmus)

Um die Anzahl der Netzwerkelemente und den Berechnungsaufwand während der Simulation zu reduzieren bietet es sich an Waypoint-Elemente nur an charakteristischen Positionen innerhalb des Straßennetzwerks zu erzeugen. Hierbei muss darauf geachtet werden, dass der Spurverlauf nicht entscheidend verfälscht wird sondern die beschriebene Kurve möglichst genau repräsentiert wird. Die Distanz innerhalb welcher Waypoint-Elemente gesetzt werden müssen hängt stark von der Krümmung des Segmentes ab. Stark gekrümmte Segmente benötigen eine größere Anzahl an Waypoint-Elementen um ihre Form beizubehalten und weniger stark gekrümmte Segmente benötigen weniger Waypoint-Elemente. Zur Berechnung von charakteristischen Positionen für die Elemente wird der Douglas-Peucker-Algorithmus [6] verwendet. Über diesen Algorithmus wird die Anzahl der berechneten Positionen minimiert, während die Form des durch die Positionen beschriebenen Kantenzugs innerhalb einer gegeben Toleranz eingehalten wird. In Abbildung 19 sind zwei Aufnahmen des selben Kreuzungsabschnitts dargestellt. In der linken Darstellung wurden Waypoint-Elemente in Distanzen von einem Meter erzeugt. In der rechten Darstellung ist die Anzahl der Elemente unter Verwendung des Douglas-Peucker-Algorithmus reduziert worden. Dabei wurde eine beispielhafte Toleranz von $\epsilon = 0,3m$ gewählt.

Schritt 2: Verbindungen setzen

Nachdem alle Lane- und Connector-Elemente erzeugt wurden, ist es möglich die Verbindungen zwischen diesen Elementen zu setzen. Dadurch, dass jedes Connector-Element ein eindeutiges Lane-Element als Vorgänger und ein eindeutiges Lane-Element als Nachfolger besitzt, reicht es aus die Verbindungen aller Connector-Elemente zu durchlaufen. Dadurch ist gewährleistet, dass auch alle Vorgänger- und Nachfolgerelemente der Lane-Elemente berücksichtigt werden.

Um die Verbindungen zu setzen werden alle Road-Einträge mit einer Referenz auf einen Junction-Eintrag – also Path-Einträge – durchlaufen. Während des Durchlaufs wird zuerst der betrachtete Path-Eintrag, dessen vorangehender Road-Eintrag und dessen nachfolgender Road-Eintrag ermittelt. Zusätzlich werden die dazugehörigen bereits erstellten Elemente und die Kindelemente ermittelt. Also das Path-Element mit den zugehörigen Connector-Elementen sowie die beiden

Road-Elemente mit ihren Lane-Elementen. Nun werden die Connector-Elemente des momentan betrachteten Pfad-Eintrags durchlaufen. Für jedes Connector-Element wird zuerst das vorangehende Lane-Element und anschließend das nachfolgende Lane-Element bestimmt.

Um das vorangehende Lane-Element zu ermitteln, werden die zum Path-Element gehörenden Lane-Einträge des ersten LaneSection-Eintrags ermittelt. Darin wird der zum Connector-Element gehörende Lane-Eintrag ermittelt. Die im darin enthaltenen Predecessor-Eintrag gespeicherte id wird vermerkt. Nun werden die vorher hinterlegten vorangehenden Lane-Elemente durchlaufen. Das Lane-Element, dass mit der ermittelten id übereinstimmt, wird im Connector-Element als Vorgänger eingetragen. Im Lane-Element wird das aktuelle Connector-Element als Nachfolger eingetragen.

Um das nachfolgende Lane-Element zu ermitteln werden die zum Path-Element gehörenden Lane-Einträge des letzten LaneSection-Eintrags ermittelt. Darin wird der zum Connector-Element gehörende Lane-Eintrag ermittelt. Die darin im darin enthaltenen Successor-Eintrag gespeicherte id wird vermerkt. Nun werden die vorher hinterlegten nachfolgenden Lane-Elemente durchlaufen. Das Lane-Element das mit der ermittelten id übereinstimmt wird im Connector-Element als Nachfolger eingetragen. Im Lane-Element wird das aktuelle Connector-Element als Vorgänger eingetragen.

Schritt 3: Informationen zu Waypoint-Elementen hinzufügen

Im dritten Schritt werden Informationen zu bestehenden Waypoint-Elementen hinzugefügt. Die jeweils letzten Waypoint-Elemente eines Elements werden mit Informationen über das nachfolgende Element angereichert. Momentan werden Waypoint-Elemente deren überliegendes Element keinen Nachfolger besitzt mit *END_OF_LANE* markiert. Waypoint-Elemente deren überliegendes Element mindestens ein Connector-Element als Nachfolger besitzt werden mit *UNREGULATED_CROSSROAD* markiert. Zukünftig kann dieser Schritt erweitert werden um außer Spurende und unregulierter Kreuzung noch andere Eigenschaften wie z.B. Ampelkreuzung oder Kreisverkehr zu hinterlegen.

Schritt 4: Zusätzliche Waypoint-Elemente erzeugen

Es ist noch ein Schritt angedacht in dem zusätzliche Waypoint-Elemente mit zusätzlichen Informationen in die bestehenden Segmente eingefügt werden sollen. In Abschnitt 3.2.3 wurden bereits einige Möglichkeiten betrachtet wie und wo solche Informationen gewonnen werden können. Zur Realisierung bietet es sich an eine Funktion zu implementieren, die an jeder Position an der Informationen durch einen der *Road Description Record* hinterlegt wurden, ein Waypoint-Element mit der entsprechenden Information erzeugt.

Schritt 5: Junction-Elemente erzeugen

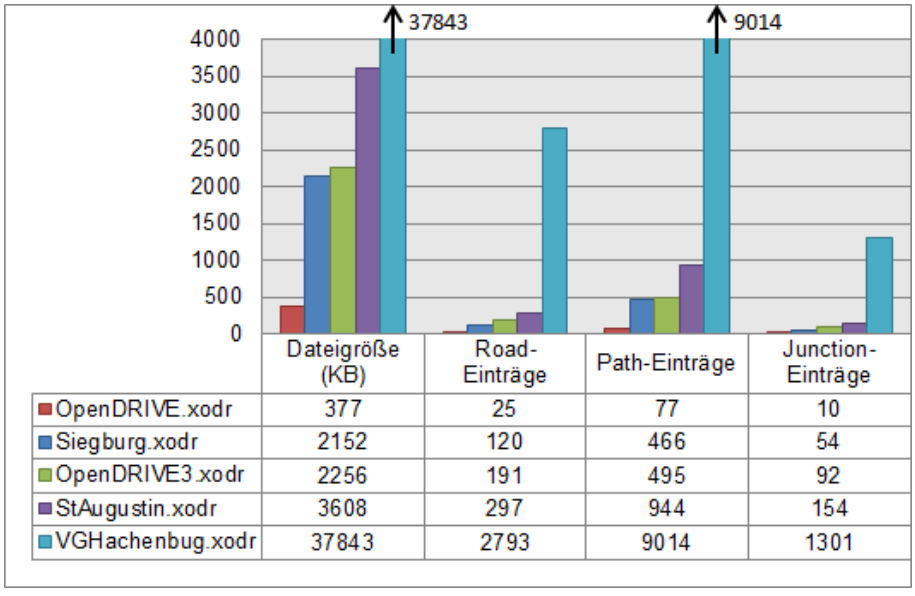
Im letzten Schritt werden Junction-Elemente erzeugt und Referenzen auf zugehörige Connector-Elemente darin hinterlegt. Als erstes werden alle Junction-Einträge durchlaufen. Zu jedem Eintrag wird ein Junction-Element erzeugt. Diesem Junction-Element werden alle zu den im Junction-Eintrag referenzierten Road-Einträgen gehörenden Connector-Elemente zugewiesen. Über die zugehörigen Connector-Elemente wird der Bereich des Junction-Elements bestimmt. Zukünftig kann realisiert werden, dass automatisch Informationen über Vorfahrtsregelungen im Junction-Element hinterlegt werden.

5. Evaluation

Dieses Kapitel befasst sich mit der Evaluation des durchgeführten Projektes. Bevor jedoch im folgenden Abschnitt die Evaluierungsstrategie erläutert wird, werden einige Informationen zu den während der Evaluation verwendeten Testdaten festgehalten. Zur Evaluation wurden mehrere OpenDRIVE®-Datensätze verwendet. Diese sind mitsamt den zugrundeliegenden Trian3D Builder Projekten in Tabelle 13 aufgelistet. In der Tabelle ist zusätzlich vermerkt ob das Projekt als Beispielprojekt im Umfang der Trian3D Builder Software enthalten war oder ob der Datensatz zusätzlich als Teil des Projektes erzeugt wurde. In Abbildung 20 sind die OpenDRIVE®-Datensätze, deren Größe sowie die Anzahl der darin enthaltenen Road- und Junction-Einträge aufgelistet. Es wurden absichtlich Datensätze mit unterschiedlichen Größen und Eigenschaften ausgewählt um so auf eine vielfältigere Testdatenbasis zurückgreifen zu können.

Trian3D Builder Projekt	**OpenDRIVE®-Datei**	**Im Trian3D Builder enthaltenes Beispiel**
detailed_city.tbp	OpenDrive.xodr	Ja
example_14_OpenDrive.tbp	OpenDrive2.xodr	Ja
boston_suburban_roads.tbp	OpenDrive3.xodr	Ja
	Siegburg.xodr	Nein
	VGHachenbug.xodr	Nein
	StAugustin.xodr	Nein

Tabelle 13: Übersicht der verwendeten OpenDRIVE®-Datensätze.



	Dateigröße (KB)	Road-Einträge	Path-Einträge	Junction-Einträge
OpenDRIVE.xodr	377	25	77	10
Siegburg.xodr	2152	120	466	54
OpenDRIVE3.xodr	2256	191	495	92
StAugustin.xodr	3608	297	944	154
VGHachenbug.xodr	37843	2793	9014	1301

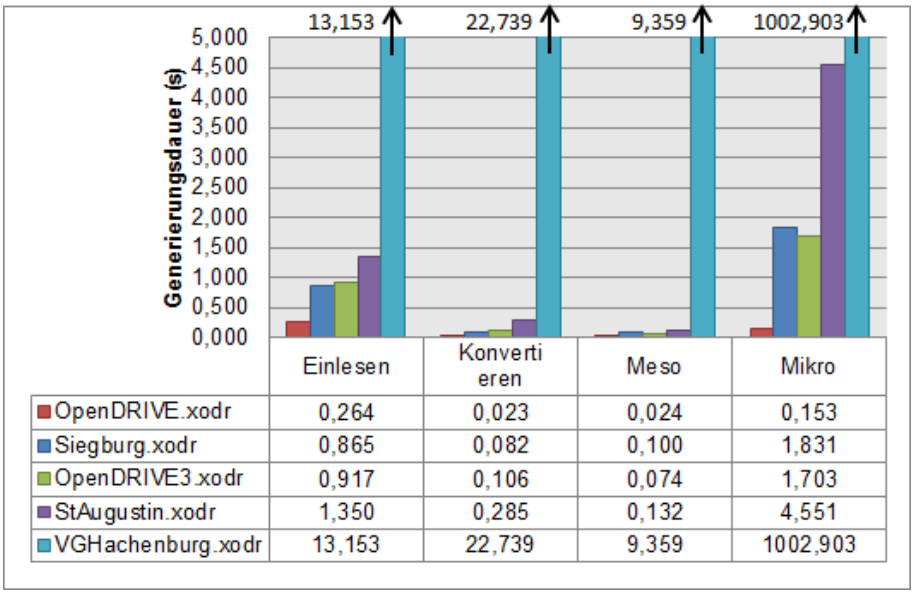
Abbildung 20: Informationen zu den fünf in der Evaluation verwendeten OpenDRIVE®-Datensätzen.

5.1. Evaluierungsstrategie

Zur Evaluation wurden mehrere sich ergänzende Evaluierungsstrategien eingesetzt. Zunächst wurden einige Leistungsmerkmale ermittelt und festgehalten. Anschließend wurden die geometrischen Berechnungen, welche als Basis für die Generierung der Verkehrsnetzwerke dienen überprüft, bevor die Netzwerke durch die aktive Navigation von Agenten auf diesen auf Nutzbarkeit hin geprüft wurden. Abschließend wurde eine Evaluation durch Beobachtung ausgeführt. Bei dieser wurde ein bisher nicht betrachtetes Evaluierungsszenario definiert und Testdaten zu diesem generiert. Aus den generierten Testdaten wurden Verkehrsnetzwerke generiert. Diese Netzwerke wurden durch einen ausgewählten Probanden mit realem Kartenmaterial verglichen und die Ergebnisse festgehalten. Die festgehaltenen Informationen wurden abschließend untersucht und

Komponententyp	Bezeichnung	Eigenschaften
CPU	Intel Core 2 Quad Q6700	4x 2.67GHz
RAM		4GB
GPU	NVIDIA GeForce 8800 Ultra	
HDD	SAMSUNG HD501LJ SCSI Disk Device	2x 465GB
OS	Windows7 Professional	

Tabelle 14: Eigenschaften des für die Evaluation verwendeten Computers.



	Einlesen	Konvertieren	Meso	Mikro
OpenDRIVE.xodr	0,264	0,023	0,024	0,153
Siegburg.xodr	0,865	0,082	0,100	1,831
OpenDRIVE3.xodr	0,917	0,106	0,074	1,703
StAugustin.xodr	1,350	0,285	0,132	4,551
VGHachenburg.xodr	13,153	22,739	9,359	1002,903

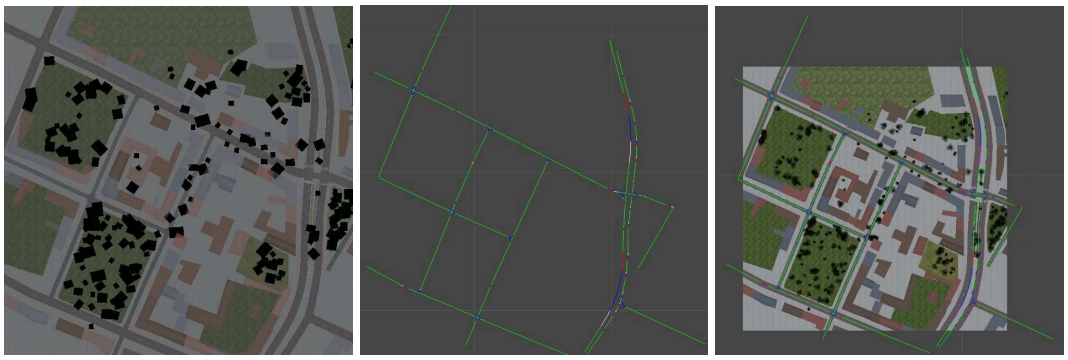
Abbildung 21: Dauer der Generierung von Verkehrsnetzwerken aus verschiedenen Datensätzen gemittelt über 10 Generierungsprozesse pro Netzwerk in Sekunden. Einlesen: Der Einleseprozess der OpenDRIVE®-Daten in die Simulationsumgebung. Konvertieren: Der Konvertierungsprozess der eingelesenen Daten in eine objektorientierte Darstellung. Meso: Generierung des Verkehrsnetzwerks für die mesoskopisch Simulationsebene. Mikro: Generierung des Verkehrsnetzwerks für die mikroskopische Simulationsebene.

diskutiert. Durch diese Vorgehensweise war es möglich, bestehende Probleme und zukünftige Optimierungsmöglichkeiten zu erkennen.

5.2. Festgehaltene Leistungsmerkmale

In diesem Abschnitt sind einige Leistungsmerkmale zum Generierungsprozess der Verkehrsnetzwerke festgehalten. Um diese zu ermitteln, wurde die Zeit für die Generierung von Verkehrsnetzwerken mehrerer Datensätze gemessen. In Tabelle 14 ist die Hardwareausstattung des für die Laufzeittests genutzten Rechners festgehalten und in Abbildung 21 sind die Zeiten festgehalten, die die Generierung der unterschiedlichen Verkehrsnetzwerke benötigte.

Es kann festgestellt werden, dass der Vorgang zum Einlesen der Daten sowie die Erzeugung des Verkehrsnetzwerks für die mikroskopische Ebene am meisten Zeit in Anspruch nehmen, während der Konviertierungsprozess der OpenDRIVE®-Daten in die objektorientierte Darstellung, sowie der Erzeugungsprozess für das Verkehrsnetzwerk der mesoskopischen Simulationsebene relativ wenig Zeit beanspruchen. Die Generierungsprozess hängt wie erwartet stark von der Dateigröße ab. Interessant ist, dass die Generierungsdauer des Netzwerks für die mikroskopische Ebene der Datei Siegburg.xodr die der Datei OpenDRIVE3.xodr übersteigt, obwohl diese größer ist. Dies



(a) (b) (c)

Abbildung 22: Überlagerung von Referenzlinien und dem zugehörigen 3D-Modell einer Stadt. (a) Szene einer Stadt. (b) Referenzlinien berechnet auf Basis von Daten welche der Stadtszene entsprechen. (c) Überlagerung der Referenzlinien und des 3D-Modells. Es ist ersichtlich, dass die Referenzlinien korrekt auf den Mittelstreifen der Straßen verlaufen.

ist offensichtlich auf interne Unterschiede zurückzuführen. So kann es sein, dass die Datei Siegburg.xodr schwieriger zu errechnete Geometrien beinhaltet oder mehr Spuren pro Straße definiert sind. Nach der Generierung der Verkehrsnetzwerke kann weitere Zeit verstreichen bevor die Simulation startet. Dies kommt dadurch zu Stande, dass für alle neu erzeugten Objekte Rückruffunktion wie z.B. *Awake* und *Start* aufgerufen werden.

5.3. Überprüfung der geometrischen Berechnungen

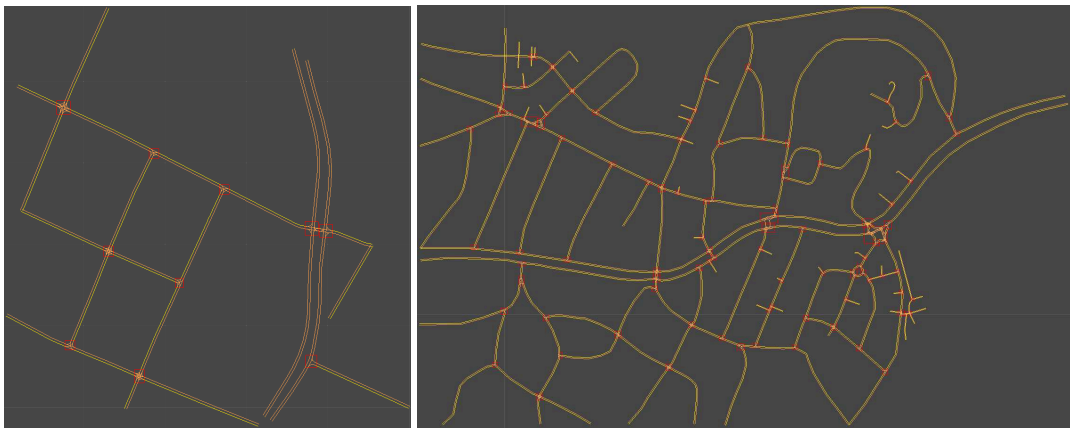
Zur Überprüfung der geometrischen Berechnungen wurde eine generische grafische Darstellung erzeugt. Mit dieser kann die Plausibilität der zur Berechnung der Straßennetzwerke verwendeten Methoden überprüft werden. Folgend werden Visualisierungen zu berechneten Straßenreferenzlinien betrachtet und mit einer zugehörigen Szene verglichen. Anschließend werden Visualisierungen von Verkehrsnetzwerken verschiedener Datensätze betrachtet.

Referenzlinien

Als erstes wurde zu den berechneten Werten für die in OpenDRIVE® definierten Straßenreferenzlinien eine Visualisierung erzeugt. In Abbildung 22 (a) ist die Draufsicht auf eine Szene, welche aus der detailied_city.tbp Datei erzeugt wurde, zu sehen. (b) zeigt die Visualisierung der in der zugehörigen OpenDRIVE®-Datei beschriebenen Referenzlinien. In (c) sind diese beiden Ansichten überlagert dargestellt. Es ist erkennbar, dass sich die berechneten Referenzlinien korrekt über die Szene legen lassen. Somit kann gezeigt werden, dass die entsprechenden Berechnungen (beschrieben in Abschnitt 2.3) korrekt erfolgen. Auf Basis der Referenzlinien war es anschließend möglich die Positionen der einzelnen Elemente des Modells für die Mikrosimulationsebene zu berechnen. Diese Elemente werden hauptsächlich auf Spuren erzeugt, welche mit einem Versatz parallel zur Referenzlinie der Straße definiert sind. Im folgenden Abschnitt werden Visualisierungen von vollständigen Verkehrsnetzwerken betrachtet.

Generierte Verkehrsnetzwerke

Es wurden Verkehrsnetzwerke zu mehreren OpenDRIVE®-Beispieldatensätzen generiert. Als erstes wurden Daten von bestehenden Beispielprojekten, welche im Umfang der Trian3D Builder Software vorhanden waren, verwendet. Hier wurden die Projekte detailed_city.tbp, boston_suburban_roads.tbp und example_14_OpenDrive.tbp genutzt. Weiterhin wurden mit Hilfe des in [12]



(a) (b)

Abbildung 23: Verkehrsnetzwerke generiert aus OpenDRIVE®-Daten welche aus bestehenden Trian3D Beispielprojekten erzeugt wurden. (a) detailed_city.tbp, (b) boston_suburban_roads.tbp.

(a) (b)

Abbildung 24: Netzwerke generiert aus OpenDRIVE®-Daten welche durch den in [12] beschriebenen Arbeitsablauf erzeugt wurden. (a) Modell zu einem Teil der Verbandsgemeinde Hachenburg. (b) Modell zu einem Teil der Stadt Siegburg.

eingeführten Arbeitsablaufes weitere Datensätze erzeugt. Der erste Datensatz beschreibt einen Teil der Stadt Siegburg. Dies ist der selbe Teil, welcher auch innerhalb des FIVIS-Simulators genutzt wird. Der zweite Datensatz repräsentiert einen Teil der Verbandsgemeinde Hachenburg. In den Abbildungen 23 und 24 sind die Visualisierungen der aus diesen Daten generierten Verkehrsnetzwerke dargestellt.

5.4. Evaluation durch Navigation von Agenten

Ziel dieses Evaluationsansatzes ist es zu zeigen, dass die generierten Verkehrsnetzwerke als Basis für die bestehende Simulation genutzt werden können. Um dies zu zeigen, wurden in generierte Verkehrsnetzwerke beider Simulationsebenen (mesoskopisch/mikroskopisch) entsprechende Agenten eingefügt und Logdateien über deren Wege innerhalb des Netzwerkes erstellt. Diese so gewonnenen Informationen wurden anschließend ausgewertet, um relevante Eigenschaften nachzuweisen.

5.4.1. Mesoskopische Simulationsebene

Zur Vereinfachung werden die mesoskopischen Netzwerke während der Evaluation als einfache gerichtete Graphen betrachtet. Node-Elemente werden dabei als Knoten und Edge-Elemente als

Kanten interpretiert. Für die Verkehrsnetzwerke der warteschlangenbasierten Mesosimulation gibt es mehrere zu evaluierende Eigenschaften. Es ist zu zeigen, dass die erzeugten Netzwerke während der Simulation keine Deadlocks verursachen. Dies wurde dadurch evaluiert, dass eine Simulation mit 100 Verkehrsteilnehmern[15] auf dem zur detailed_city.tbp Datei gehörenden Graphen 10 Stunden lang ausgeführt wurde. Bei jedem Wechsel eines Verkehrsteilnehmers von einer Kante des Graphen in eine andere wurde ein Logeintrag erstellt. Festgehalten wurden die folgenden Parameter: Zeitstempel, ID des Verkehrsteilnehmers, Ausgangskante, verbindender Knoten, Zielkante und die Wartezeit. Die anhand der Logdatei ermittelten Eigenschaften sind in Tabelle 15 aufgeführt.

Parameter	**Ermittelter Wert**
Maximale Wartezeit:	110,46s
Durchschnittliche Wartezeit:	2,86s
Durchschnittliche Wartezeit (ohne Wartezeiten von 0s):	6,88s
Durchschnittliche Anzahl an Abbiegevorgängen pro Verkehrsteilnehmer:	3641,94
Maximale Anzahl an Abbiegevorgängen pro Verkehrsteilnehmer:	3713
Minimale Anzahl an Abbiegevorgängen pro Verkehrsteilnehmer:	3564
Spannweite der Abbiegevorgänge ($max - min$):	149
Maximale Anzahl der Nutzung eines Node-Elements:	30121
Minimale Anzahl der Nutzung eines Node-Elements:	15992

Tabelle 15: Auf Basis eines 10 Stunden langen Testlaufs ermittelte Eigenschaften zur mesoskopischen Simulationsebene.

Auswertung

- *Maximale Wartezeit*
  Die durch das Logging ermittelte maximale Wartezeit liegt bei 110,46 Sekunden also ungefähr 2 Minuten. Dies schließt aus, dass eine längere Wartezeit vor einem Abbiegevorgang existierte. Jedoch könnte es sein, dass ein Verkehrsteilnehmer weiterhin in einer Wartesituation verharrt und die so entstandene Wartezeit nicht in den Logdaten auftaucht. Dass dieser Zustand nicht existiert, kann über die im folgenden ausgewerteten Abbiegevorgänge pro Verkehrsteilnehmer gezeigt werden.

- *Abbiegevorgänge pro Verkehrsteilnehmer*
  Durch die im Verhältnis gesehen geringe Spannweite und die noch geringere Distanz zwischen den Quartilen (siehe Abbildung 26), ist erkennbar, dass jeder Verkehrsteilnehmer ungefähr gleich viele Abbiegevorgänge ausgeführt hat. Daraus kann gefolgert werden, dass sich kein Verkehrsteilnehmer über eine längere Zeit hinweg in einer Engpass bzw. Deadlocksituation befunden hat.

- *Durchschnittliche Wartezeit*
  Aus der Logdatei wurde eine durchschnittliche Wartezeit[16] von 6,88 Sekunden ermittelt. Die Wartezeit resultiert aus den 3 sekündlichen Wechselintervallen der Knoten in Verbindung mit dem durchschnittlichen Knotenausgangsgrad des Gesamtnetzwerks, welcher im herangezogenen Beispiel 3,2 beträgt. In Abbildung 25 ist die Anzahl der Wartevorgänge in Zeitintervallen von jeweils 5 Sekunden dargestellt. Zu erkennen ist ein linearer Abfall der Anzahl an Wartevorgängen bei einer logarithmischen Verteilung. Das heißt, dass die Wahrscheinlichkeit, dass ein Verkehrsteilnehmer weiterhin warten muss wie zu erwarten

[15] Maximal wären bei einer ermittelten Verkehrsnetzwerkslänge von 3736,61 Metern ungefähr 500 Verkehrsteilnehmer möglich.
[16] Werte mit einer Wartezeit von 0s ausgenommen.

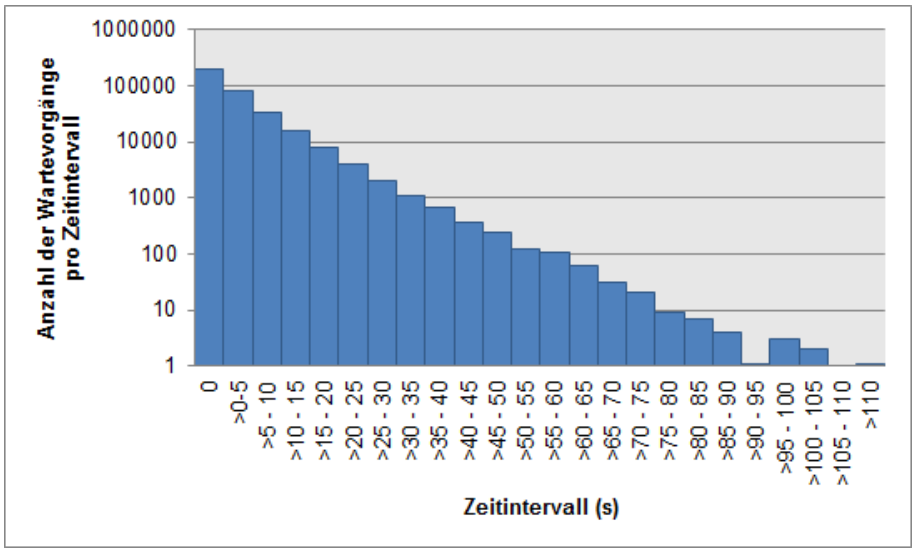


Abbildung 25: Histogramm, welches die Häufigkeitsverteilung von Wartevorgängen von Verkehrsteilnehmern in einem Netzwerk der mesoskopischen Simulationsebene darstellt. Jeder Balken repräsentiert einen Zeitintervall von 5 Sekunden. Zugrunde liegt eine 10 Stunden lange Simulation mit 100 Verkehrsteilnehmern.

exponentiell abfällt. Die durchschnittliche Wartezeit resultiert fast ausschließlich aus den Werten der ersten Intervalle.

- *Nutzung von Kreuzungen*
  Bei der Nutzung von Kreuzungen ist ersichtlich, dass Knoten mit größeren Eingangsgraden häufiger genutzt wurden als diejenigen mit niedrigen Eingangsgraden.

5.4.2. Mikroskopische Simulationsebene

Um zu zeigen, dass generierte mikroskopische Verkehrsnetzwerke als Basis für die bestehende Simulation genutzt werden können, werden Agenten in die Netzwerke eingefügt und Logdateien über deren Interaktion mit den Netzwerken erstellt. Diese werden anschließend ausgewertet. Für diese Evaluation wird eine bestimmte Klasse von Agenten genutzt. Diese wird folgend erläutert.

Positionsagenten

Die zur Evaluation verwendete Klasse von Agenten wird *Positionsagenten* (Position-Agents) genannt. Die Klasse der Positionsagenten wurde als Überführungsglied zwischen der Mikro- und Mesosimulationsebene implementiert. Sie eignen sich an dieser Stelle besonders durch die Eigenschaft, dass sie zwar auf dem mikroskopischen Netz agieren, jedoch nicht von der simulierten Physik beeinflusst werden. Der Vorteil ist, dass die Agenten nicht durch Kollisionen oder andere physikalische Effekte behindert werden, wodurch sie sonst, ohne Zutun der hier zu evaluierenden generierten Verkehrsnetzwerke, stehen bleiben könnten.

Ausführung

Die generierten Verkehrsnetzwerke besitzen auf Grund der zugrunde liegenden Daten eine Menge an unabgeschlossenen Lane-Elementen. Das letzte Waypoint-Element von Lane-Elementen, die

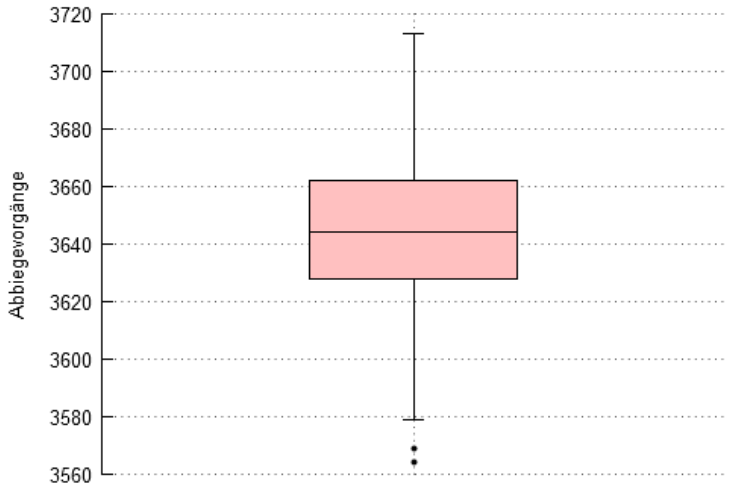


Abbildung 26: Verteilung der Anzahl an Abbiegevorgängen pro Verkehrsteilnehmer. Die Anzahl schwankt zwischen maximal 3713 und minimal 3564 Abbiegevorgängen. Zugrunde liegt eine 10 Stunden lange Simulation mit 100 Verkehrsteilnehmern.

kein Nachfolgerelement besitzen, wird jeweils als Endelement markiert und das erste Waypoint-Element von Lane-Elementen, die kein Vorgängerelement besitzen, wird als Startelement markiert. Zur Ausführung der Evaluierung wurde ein Modul implementiert das eine gegebene Anzahl an Positionsagenten in das Netzwerk einfügt. Dies geschieht dadurch, dass an einer zufällig bestimmten Teilmenge der Start-Waypoint-Elemente ein Positionsagent eingefügt wird. Wenn mehr Agenten erzeugt werden sollen als Start-Waypoint-Elemente existieren, wird eine angegebene Zeit lang gewartet und der Prozess wiederholt. Somit ist gewährleistet, dass sich nach kurzer Zeit die gewünschte Anzahl von Agenten im Netzwerk befinden. Zusätzlich besteht die Problematik, dass ein Agent ein End-Waypoint-Element erreichen kann. In diesem Fall wird der Agent zu einem zufällig gewählten Start-Waypoint-Element verschoben. Damit ist gewährleistet, dass ein gewisser Fluss im Netzwerk aufrechterhalten wird. Nachträglich wurden zwei Optimierungen am Verhalten der Positionsagenten realisiert. Positionsagenten werden nun nur noch an Start-Waypoint-Elementen eingefügt, die erstens nicht mit *shoulder* also Seitenstreifen markiert sind und zweitens kein Nachbar-Lane-Element gleicher Fahrtrichtung mit Vorgängerelement besitzen. Zusätzlich wurde ein Spurwechsel bei den Positionsagenten eingeführt, dieser wird ausgeführt wenn ein Positionsagent ein End-Waypoint-Element erreicht, jedoch ein Nachbar-Lane-Element der selben Fahrtrichtung existiert welches Nachfolgerelemente besitzt. Somit werden unnötige Neueinfügevorgänge verhindert.

Zusätzlich wurden Logger implementiert, um die Wege der Positionsagenten durch das Netzwerk festhalten zu können. Während der Simulation werden zwei verschiedene Logdateien erzeugt. In der ersten Logdatei werden die zurückgelegten Wege der Positionsagenten gespeichert. Das heißt bei jedem Wechsel zu einem neuen Element wird der Zeitpunkt, der Name des Agenten, das alte Road- bzw. Path-Element und Lane- bzw. Connector-Element und das neue Road- bzw. Path-Element und Lane- bzw- Connector-Element gespeichert. In der zweiten Logdatei wird vermerkt wenn ein Agent durch das erreichen eines End-Waypoint-Elementes zu einem Start-Waypoint-Element verschoben wird. Dazu wird ein Zeitstempel, der Name des Agenten und die Zeit vermerkt, die er im Netzwerk agiert hat, ohne dass er zu einem neuen Start-Waypoint-Element verschoben werden musste.

Ergebnisse und Auswertung

Auf dem zur Datei StAugustin.xodr generierten Verkehrsnetzwerk wurde eine Simulation von mehr als 5 Tagen ausgeführt. Das generierte Netzwerk beinhaltet 115 genutzte Start-Waypoint-Elemente und 135 End-Waypoint-Elemente. Es folgt die Diskussion der Ergebnisse.

- *Absturz / Exception freie Simulation*
  Als erstes kann festgestellt werden, dass die Simulation über eine längere Dauer ausgeführt werden kann ohne das Probleme wie z.B. Abstürze, Exceptions oder Fehlermeldungen auftreten. Damit ist gezeigt, dass die generierten Verkehrsnetzwerke keine internen Fehler aufweisen und das die von den Agenten genutzten internen Parameter entsprechend korrekt gesetzt sind.

- *Ausgewogene Nutzung der Netzwerkelemente*
  Durch die Auswertung der Daten in der ersten Logdatei kann gezeigt werden, dass alle generierten Road- sowie Lane-Elemente innerhalb des Netzwerkes genutzt wurden. Wichtig ist hierbei, dass auch eine Ausgeglichene Nutzung erfolgt. In Abbildung 27 ist die Anzahl der Befahrungen der einzelnen Road- und Lane-Elemente aufgetragen. Es ist erkennbar, dass der Großteil der zweispurigen Road-Elemente etwas weniger als 30000 mal befahren wurde wobei sich die Befahrungen gleichmäßig auf beide Lane-Elemente aufteilen. Sämtliche Elemente die eine andere Charakteristik aufweisen stellen Sonderfälle dar. Im mittleren Teil der Grafik ungefähr bei Road-Element 11855 und im rechten Teil der Grafik bei Road-Element 12095 sind die Road-Elemente der Kreisverkehre zu finden welche natürlich nur eine Fahrtrichtung und damit nur ein Lane-Element besitzen. Diese wurden ebenfalls etwas weniger als 30000 mal befahren. Sämtliche wenig befahrenen Elemente[17] repräsentieren Straßen die mit dem Autobahnabschnitt in Verbindung stehen. Die fünf dreipurigen Elemente repräsentieren die Autobahnabschnitte. Die restlichen mehrspurigen Abschnitte befinden sich auf der Hauptstaße die vom Autobahnabschnitt nach Süden durch die Stadt führt. In der Gesamtheit betrachtet kann eine ausgewogene Nutzung der Netzwerkelemente festgestellt werden.

- *Unterbrechungsfreie Navigation der Agenten*
  Durch die Auswertung der Daten in der zweiten Logdatei kann gezeigt werden, dass es Agenten möglich ist sich über eine längere Zeit im Netzwerk zu bewegen ohne durch erreichen eines End-Waypoint-Elementes verschoben werden zu müssen, also dass ein gewisser Zusammenhang besteht. In Abbildung 28 ist die Anzahl der Verweilvorgänge von Agenten in Intervallen von 10 Sekunden aufgetragen. Die Werte sind einmal linear (oben) und einmal logarithmisch skaliert (unten) dargestellt. Es ist erkennbar, dass ein annähernd exponentieller Abfall der Anzahl der Verweildauern besteht. Dies liegt an der zufälligen Wegwahl der Positionsagenten. Die vielen kurzen Verweildauern kommen durch offene Enden am Rand des Netzwerkes aber z.B. auch in Wohngebietsbereichen zu Stande. Wichtiger ist jedoch, dass auch längere Verweildauern (bis zu 23 Minuten) existieren. Daraus kann gefolgert werden, dass das generierte Netzwerk auch für die Überwindung langer Strecken hinreichend verbunden ist. Die zwei erkennbaren Außreißer kommen mit hoher wahrscheinlichkeit durch Charakteristiken des genutzten Netzwerks zu stande.

5.5. Evaluation durch Beobachtung

Als letztes soll auf Basis von Beobachtung eine Evaluation der Qualität der erzeugten Verkehrsnetzwerkmodelle vollzogen und gleichzeitig bestehende Probleme erkannt werden. Im folgenden Abschnitt wird die Evaluierungsstrategie beschrieben. Anschließend wird die Durchführung be-

[17] Nahe 11674, 11780, 11872, 11958 und 12031.

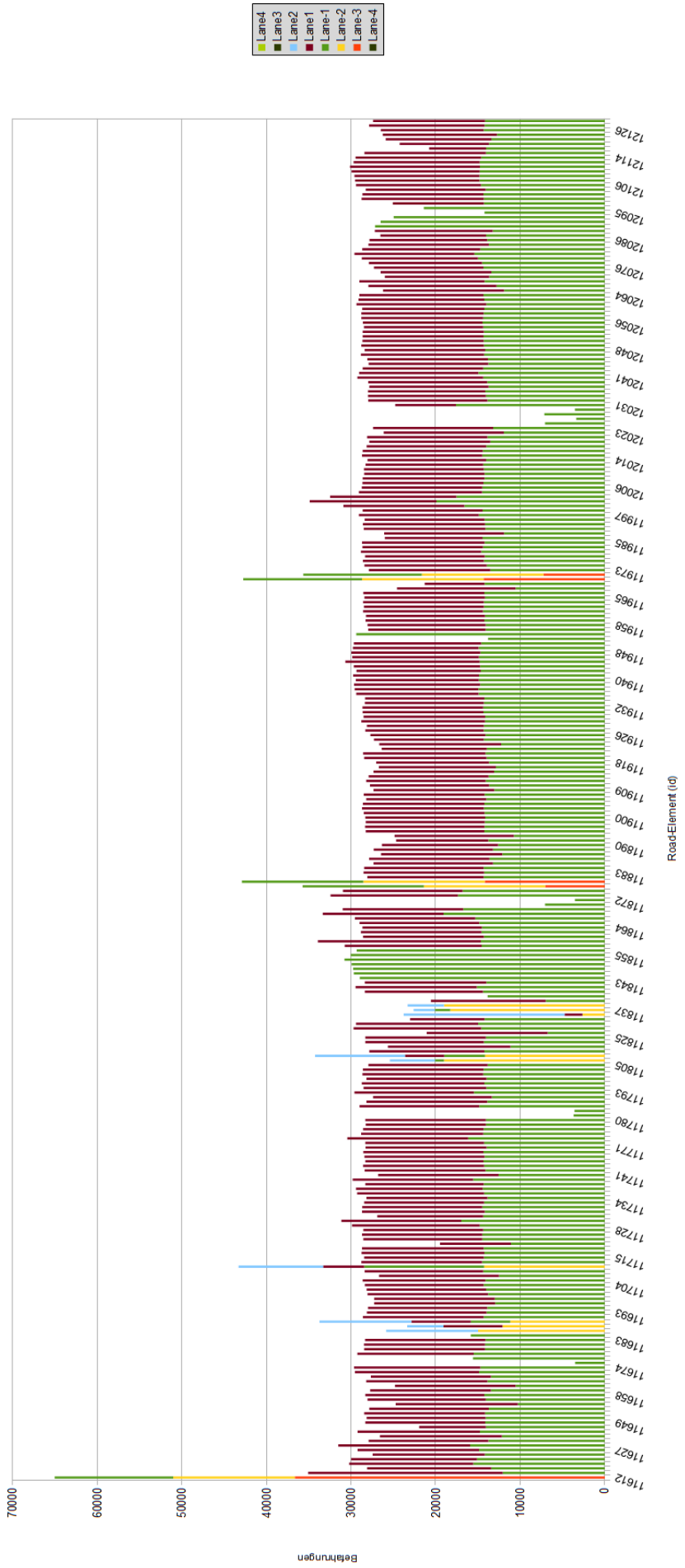


Abbildung 27: Anzahl der Befahrungen einzelner Road- und Lane-Elemente während eines Testlaufes auf einem Verkehrsnetzwerk der mikroskopischen Ebene.

Abbildung 28: Histogramm, dass die Häufigkeitsverteilung von Verweilzeiten der Agenten in einem Netzwerk der mikroskopischen Simulationsebene darstellt. Jeder Balken repräsentiert die Anzahl der Zeiten innerhalb eines zehnsekündigen Zeitintervalls. Oben in linearer Skalierung und unten in logarithmischer Skalierung aufgetragen.

schrieben und die gewonnen Informationen protokolliert. Letztendlich werden die gewonnenen Informationen ausgewertet.

5.5.1. Evaluierungsstrategie und Kriterien

Die Evaluation verläuft in mehreren Schritten. Als erstes soll ein OpenDRIVE®-Datensatz zu einem bisher noch nicht betrachteten Szenario erzeugt werden. Um die Qualität der Evaluation zu erhöhen, soll das gewählte Szenario bestimmte Kriterien erfüllen. Die Kriterien werden im folgenden Abschnitt aufgelistet. Aus dem erzeugten OpenDRIVE®-Datensatz sollen anschließend die Verkehrsnetzwerke für beide Simulationsebenen generiert werden. Anschließend soll das detailliertere, also das mikroskopische Verkehrsnetzwerk, auf verschiedene Weisen betrachtet werden. Da das mesoskopische Verkehrsnetzwerk eine Abstraktion des mikroskopischen Netzwerks darstellt wird dieses in diesem Kapitel nicht weiter betrachtet. Als erstes soll ein Abgleich mit Landkartendaten erfolgen. Hier bietet sich die OpenStreetMap[18] Karte an, welche auf den gleichen Daten basiert die auch zur Erzeugung der OpenDRIVE®-Datensätze verwendet wurde. Um einen Vergleich machen zu können, kann die passende Google Maps[19] Karte herangezogen werden. Der Abgleich soll bestenfalls durch eine dritte, mit dem Projekt nur wenig vertraute, Person erfolgen. Die Person soll zuerst einen allgemeinen Vergleich der erzeugten Netzwerke und des Kartenmaterials durchführen. Anschließend sollen einige ausgewählte Abschnitte des Szenarios genauer untersucht werden. Bei Unklarheiten oder Aspekten die aus dem Kartenmaterial nicht hervorgehen, kann gegebenenfalls ein Vergleich mit der echten Straße erfolgen (unter der Voraussetzung das der betrachtete Ort ohne allzu großen Aufwand erreichbar ist). Letztendlich wird eine Auswertung der durch die dritte Person gewonnen Informationen durchgeführt. Durch diese sollen die Grenzen des umgesetzten Generierungsprozesses festgestellt werden. Idealerweise soll festgestellt werden an welcher Stelle sich im in [12] eingeführten Arbeitsablauf noch Schwachstellen befinden bzw. wo wichtige Informationen verloren gehen oder entsprechend ergänzt werden müssen.

Auswahlkriterien für das Szenario

Die in der folgenden Liste aufgeführten Kriterien sollen bei der Auswahl des zur Evaluierung verwendeten Szenarios beachtet werden. Dadurch soll gewährleistet werden, dass ein repräsentatives Szenario gewählt wird, welches verschiedenste Aspekte beinhaltet und somit eine objektive Einschätzung erlaubt.

- Das Szenario soll allen beteiligten Personen hinreichend bekannt sein, damit die Plausibilität der erzeugten Netzwerke entsprechend geprüft werden kann.
- Das Szenario soll für die Testperson ohne größeren Aufwand erreichbar sein. So soll gewährleistet sein, dass im Zweifelsfall ein Abgleich mit der Realität erfolgen kann.
- Um eine repräsentative Auswahl zu gewährleisten soll das Szenario einige Merkmale enthalten, die für Verkehrsnetzwerke typisch sind:
  - Im Szenario sollen verschieden große Kreuzungen enthalten sein.
  - Im Szenario soll mindestens ein mehrspuriger Straßenabschnitt oder ein Autobahnabschnitt enthalten sein.
  - Im Szenario soll mindestens ein Kreisverkehr enthalten sein.

[18] `www.openstreetmap.de/karte.html`
[19] `http://maps.google.de/`

– Im Szenario soll mindestens ein Abschnitt enthalten sein bei dem sich zwei Straßen auf verschiedenen Ebenen kreuzen, z.B. eine Brücke oder eine Unterführung.

– Im Szenario sollen Fahrradwege enthalten sein.

Fragekatalog

Folgende Liste mit Fragen soll während des Vergleichs der Daten durch die dritte Person als Leitfaden dienen.

- Stimmt der Straßenverlauf des Netzwerksausschnitt mit dem auf dem realen Kartenausschnitt überein?
- Stimmt die Anzahl der Spuren der Straßen im Netzwerksausschnitt mit denen auf dem realen Kartenausschnitt überein?
- Sind im Netzwerksausschnitt alle Verzweigungen vorhanden oder existieren im Kartenausschnitt zusätzliche Verzweigungsmöglichkeiten?
- Falls der Netzwerksausschnitt eine Verzweigung besitzt, sind die Richtungen in denen an einer Verzweigung ein Weiterfahren möglich ist korrekt?
- Können weitere erwähnenswerte Besonderheiten festgestellt werden?

Wahl des Szenarios und Generierung der OpenDRIVE®-Daten

Als Szenario wurde ein Gebiet gewählt, welches einen Teil der Stadt Sankt Augustin beinhaltet. Es erstreckt sich zwischen den folgenden Koordinaten:

$50°46'39,00''N - 50°47'22,97''N$
$7°10'46,38''O - 7°12'28.94''O$

Zur Auswahl gab es mehrere Gründe. Das Szenario betrachtet einen Bereich nahe der Hochschule Bonn-Rhein-Sieg, welcher allen beteiligten Personen bekannt ist. Weiterhin sind verschiedenste Aspekte enthalten, darunter ein Autobahnabschnitt mit darüber führender Brücke, Wohngebiet, Gewerbegebiet, Ampeln, Kreisverkehre und Fahrradwege. In Abbildung 29 ist der Ausschnitt des Szenarios innerhalb von OpenStreetMap dargestellt. Für die geforderten Merkmale steht eine Legende zur Verfügung. Zum Vergleich ist in Abbildung 30 der Abschnitt aus der Google Maps Karte zu sehen.

Zur Erzeugung von OpenDRIVE®-Daten zum gewählten Szenario wurde der in [12] beschriebene Arbeitsablauf genutzt. Dazu wurde das Gebiet zwischen den oben beschriebenen Koordinaten aus einem OpenStreetMap Shapefile des Bundeslandes Nordrhein-Westfalen[20] in die Trian3D Builder Software importiert. Die Software ermöglicht es OpenDRIVE®-Dateien zu generieren, dazu sind mehrere Schritte notwendig. Die Schritte können im Dokument *Trian3D Builder Roads Module –Workflow and Features in v4.0–* nachgelesen werden. Dort ist zusätzlich ein Tutorial vorhanden. TrianGraphics stellt auch Videobeispiele für diesen Prozess zur Verfügung[21].

[20] Verfügbar unter: `http://download.geofabrik.de/europe/germany.html`
[21] z.B. unter: `www.triangraphics.de/videos`

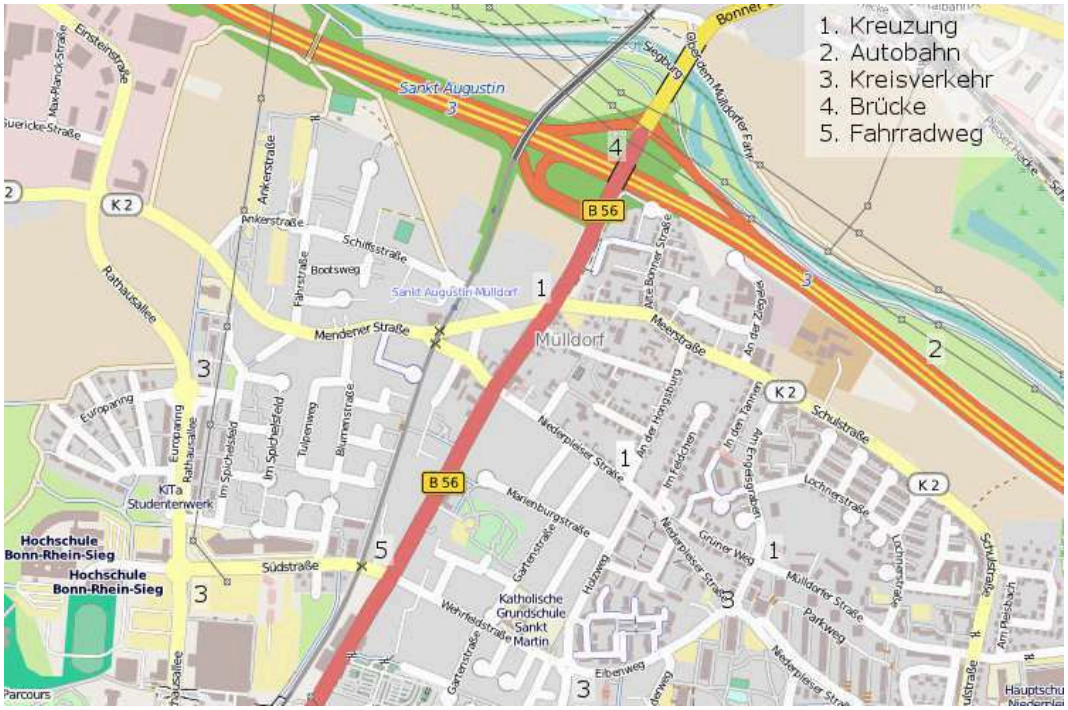


Abbildung 29: Zur Evaluation gewähltes Szenario. Das Szenario beinhaltet einen Teil der Stadt Sankt Augustin. Unter anderem ist die Hochschule Bonn-Rhein-Sieg enthalten. Bild basierend auf OpenStreetMap Kartenausschnitt: `www.openstreetmap.de/karte.html`.

Abbildung 30: GoogleMaps Kartenausschnitt zum Evaluierungsszenario. Das Szenario beinhaltet einen Teil der Stadt Sankt Augustin. Quelle: `https://maps.google.de`.

Vorbereitung der Evaluation

Als Vorbereitung zur Evaluation wurden die Verkehrsnetzwerke beider Simulationsebenen aus den OpenDRIVE®-Daten generiert. In den Abbildungen 31 und 32 sind die beiden Verkehrsnetzwerke dargestellt. Um zu Beginn der Evaluation einen Abgleich der Straßenverläufe des generierten Netzwerks, mit den beiden Karten durchführen zu können wurde der OpenStreetMap und der Google Maps Kartenausschnitt mit dem mikroskopischen Verkehrsnetzwerk überlagert. In den Abbildungen 31 bis 34 sind die generierten Netzwerke sowie die Überlagerungen des mikroskopischen Netzwerks mit dem Kartenmaterial dargestellt.

5.5.2. Durch Beobachtung gewonnene Informationen

Zur Ausführung wurde ein nicht unmittelbar mit dem Projekt vertrauter Proband ausgewählt. Der Proband wurde in das im vorigen Abschnitt eingeführte Szenario eingewiesen. Beginnend wurden die Kartenabschnitte als Ganzes mit den generierten Netzwerken verglichen. Dazu dienten unter anderem die Abbildungen 33 und 34 in denen Verkehrsnetzwerk und Kartenmaterial überlagert dargestellt sind. Anschließend wurden einzelne Aspekte des Szenarios betrachtet. Während der Ausführung wurden Notizen zu den Beobachtungen und Anmerkungen der Probanden gemacht[22], welche in den folgenden Abschnitten ausführlich dokumentiert sind und anschließend diskutiert werden.

Vergleich des generierten Verkehrsnetzwerks mit Kartenmaterial

Der Proband hat festgestellt, dass das generierte Verkehrsnetzwerk im allgemeinen korrekt aussieht. Es befindet sich an keiner Stelle im Verkehrsnetzwerk eine Straße an der im Kartenmaterial keine Straße existiert. Der umgekehrte Fall trifft jedoch nicht zu, so gibt es mehrere Bereiche in denen kleinere Straßen existieren die nicht im Verkehrsnetzwerk vorhanden sind.

Vergleich ausgewählter Aspekte

Es folgt die Betrachtung einzelner Aspekte. Fünf Bereiche innerhalb des gewählten Szenarios wurden einer genaueren Betrachtung durch den Probanden unterzogen. Die Ergebnisse sind folgend erläutert und diskutiert. Bereich 1 und 2 stellen jeweils einen Kreuzungsabschnitt dar, Bereich 2 und 3 einen Abschnitt mit einem Kreisverkehr und Bereich 5 beinhaltet ein Stück einer Autobahn mit Auf- und Abfahrten sowie einer Brücke über die Autobahn. In Abbildung 35 ist die Lage der gewählten Bereiche innerhalb des Szenarios dargestellt.

Bereich 1 und Bereich 2: Straßen und Kreuzungen

Der erste Bereich beinhaltet die Vierwegekreuzung an der die Südstraße und die Wehrfeldstraße in die B56/Bonner Straße münden. In Abbildung 36 ist Kartenmaterial sowie das generierte Netzwerk zu Bereich 1 dargestellt. Folgend sind die Feststellungen des Probanden aufgelistet:

- Der Straßenverlauf des Netzwerksausschnittes entspricht dem realen Straßenverlauf.
- Die Südstraße sowie die Wehrfeldstraße sind mit jeweils zwei Spuren korrekt dargestellt.
- Die B56/Bonner Straße scheint problematisch. Sie besitzt vier Spuren anstelle von zwei. Jeweils eine der erzeugten Spuren besitzt vor bzw. nach der Kreuzung keine Verbindung.

[22] Hier sollte beachtet werden, dass Probanden generell eher negative Aspekte erkennen und das positive Aspekte oft als selbstverständlich hingenommen werden.

Abbildung 31: Zum Evaluierungsszenario generiertes Verkehrsnetzwerk der mikroskopischen Simulationsebene. Das Szenario beinhaltet einen Teil der Stadt Sankt Augustin.

Abbildung 32: Zum Evaluierungsszenario generiertes Verkehrsnetzwerk der mesoskopischen Simulationsebene. Das Szenario beinhaltet einen Teil der Stadt Sankt Augustin. Da ersichtlich ist, dass das Netzwerk der mesoskopischen Ebene eine Abstraktion des Netzwerks der mikroskopischen Ebene darstellt, wird es in diesem Abschnitt nicht weiter betrachtet.

Abbildung 33: OpenStreetMap Kartenausschnitt überlagert mit dem generierten Verkehrsnetzwerk. Zu erkennen ist, dass das Netzwerk sehr genau mit der Karte übereinstimmt. Da der Karte und dem generierten Verkehrsnetzwerk die gleichen Daten zu Grunde liegen, war dies zu erwarten. Bild basierend auf OpenStreetMap Kartenausschnitt: `www.openstreetmap.de/karte.html`.

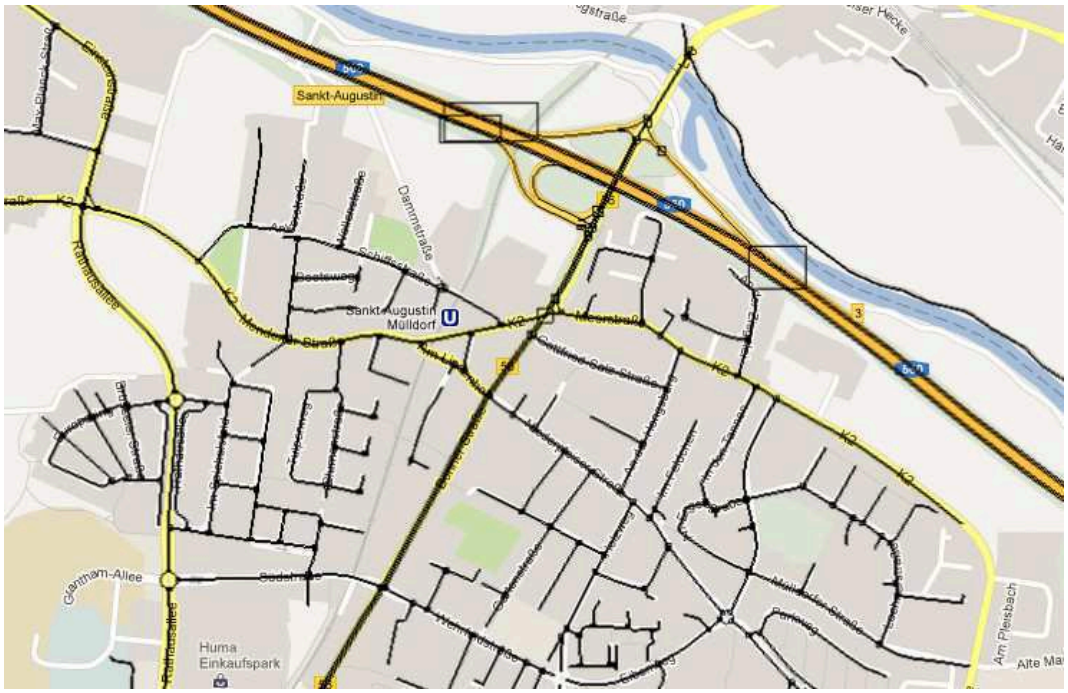


Abbildung 34: GoogleMaps Kartenausschnitt überlagert mit dem generierten Verkehrsnetzwerk. Es können kleinere Abweichungen festgestellt werden was jedoch auf die unterschiedliche Datenbasis zurückgeführt werden kann. Bild basierend auf Google Maps Kartenausschnitt: `https://maps.google.de`.

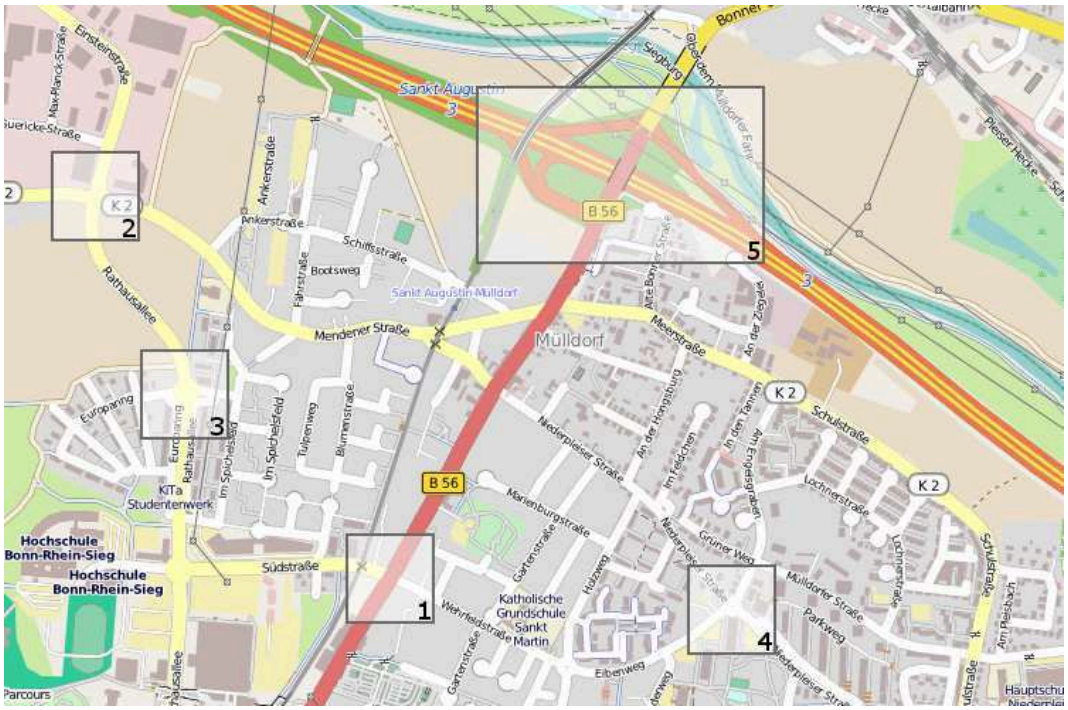


Abbildung 35: Die umrandeten Gebiete welche mit 1 bis 5 nummeriert sind, stellen die näher untersuchte Bereiche innerhalb des Szenarios dar. Bild basierend auf OpenStreetMap Kartenausschnitt: `www.openstreetmap.de/karte.html`.

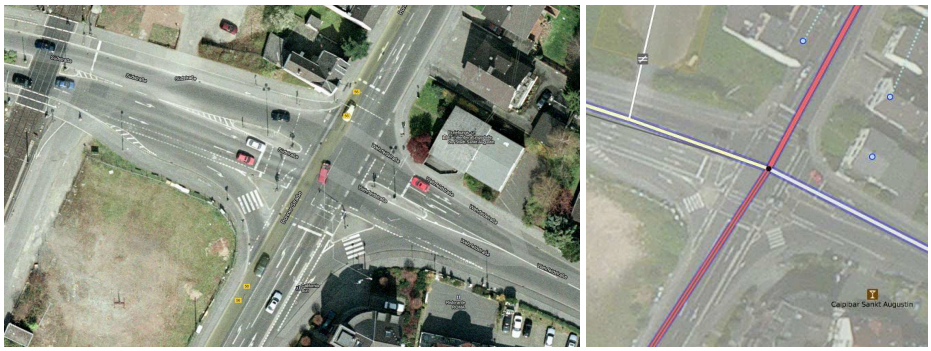
- Auf der Kreuzung scheint eine Verschiebung der anderen drei Spuren stattzufinden.
- Im Kreuzungsbereich existiert für jede ankommende Straße in jede Fahrtrichtung ein Pfad, was dem realen Straßenverlauf entspricht.
- Die verschiedenen Abbiegespuren der Straßen sind im generierten Netzwerk nicht vorhanden.

Der zweite Bereich beinhaltet die Vierwegekreuzung im Nordwesten des Szenarios. Die Siegburger Straße, Einsteinstraße, Rathausalle und die K2/Mendener Straße treffen dort aufeinander. In Abbildung 37 ist Kartenmaterial sowie das generierte Netzwerk zu Bereich 2 dargestellt. Folgend sind die Feststellungen des Probanden aufgelistet:

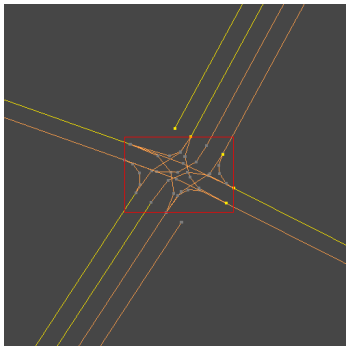
- Der Straßenverlauf des Netzwerksausschnittes entspricht dem realen Straßenverlauf.
- Die verschiedenen Abbiegespuren der Straßen sind wie auch schon im ersten Bereich des Netzwerks nicht vorhanden.
- Die Ausweichspur, welche von Osten nach Norden an der Kreuzung vorbei führt und zur Umgehung der Kreuzung dient, ist im Verkehrsnetzwerk explizit vorhanden.
- Es existieren jedoch *verbotene* Wege auf diese Ausweichspur. Das heißt, im Verkehrsnetzwerk existiert ein Pfad über den man aus Richtung der Kreuzung kommend auf die Ausweichspur auffahren kann.

Bereich 3 und Bereich 4: Kreisverkehre

Der dritte und vierte Bereich beinhaltet jeweils einen Kreisverkehr wovon sich einer im Westen und der andere im Südosten des Szenarios befinden. In den Kreisverkehr aus Bereich 3 münden



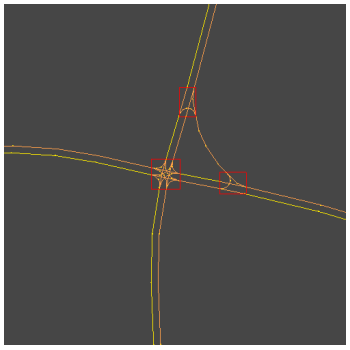
(a) (b)



(c)

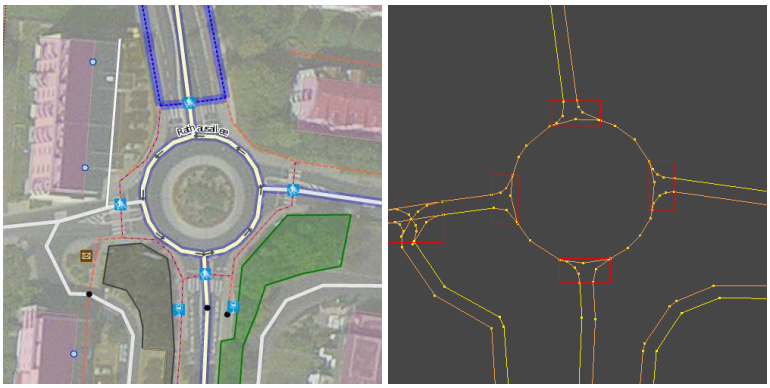
Abbildung 36: Szenario Bereich 1. Dargestellt ist eine Kreuzung im Süden des Szenarios. (a) Kreuzungsausschnitt aus Google Maps. (b) Kreuzungsausschnitt aus OpenStreetMap. (c) Verkehrsnetzwerk zum Kreuzungsausschnitt.

(a) (b)



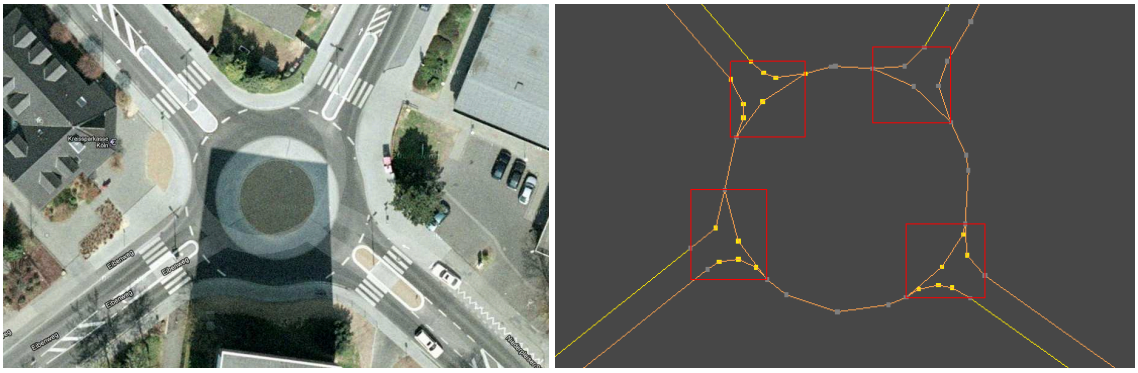
(c)

Abbildung 37: Szenario Bereich 2. Dargestellt ist eine Kreuzung im Nordwesten des Szenarios. (a) Kreuzungsausschnitt aus Google Maps. (b) Kreuzungsausschnitt aus OpenStreetMap. (c) Verkehrsnetzwerk zum Kreuzungsausschnitt.



(a) (b)

Abbildung 38: Szenario Bereich 3. Dargestellt ist ein Kreisverkehr im Westen des Szenarios. (a) Kreuzungsausschnitt aus OpenStreetMap. (b) Verkehrsnetzwerk zum Kreisverkehr. Der Google Maps Kartenausschnitt ist in dieser Abbildung nicht dargestellt, da er veraltet ist.



(a) (b)

Abbildung 39: Szenario Bereich 4. Dargestellt ist ein Kreisverkehr im Südosten des Szenarios. (a) Kreuzungsausschnitt aus Google Maps. (b) Verkehrsnetzwerk zum Kreisverkehr.

die Straßen: Rathausallee, Europaring und Im Spichelsfeld. In den Kreisverkehr aus Bereich 4 münden die Straßen: Niederpleiser Straße, Am Engelsgraben und Eibenweg. In den Abbildungen 38 und 39 ist Kartenmaterial sowie die generierten Netzwerke zu Bereich 3 und 4 dargestellt. Folgend sind die Feststellungen des Probanden aufgelistet:

- Die Straßenverläufe der Netzwerksausschnitte entsprechen den realen Straßenverläufen.
- Der Aufbau der Kreisverkehre erscheint plausibel und wirklichkeitsgetreu[23].
- Es gibt keine Unterscheidung zwischen Kreisverkehren und Verzweigungen bzw. Kreuzungen. Falls zukünftig etwas wie ein Navigationssystem für die Verkehrsteilnehmer implementiert werden sollte, kann ohne weiteren Aufwand keine Unterscheidung zwischen Kreuzung und Kreisverkehr getroffen werden also "Die dritte Abfahrt im Kreisverkehr nehmen" anstelle von "Jetzt bitte rechts abbiegen".

[23] Hier ist anzumerken, dass der Aufbau der Kreisverkehre äquivalent zum beschriebenen Aufbau innerhalb der OpenDRIVE®-Spezifikation [7, S.81] ist.

Bereich 5: Autobahn, Autobahnauffahrt/-abfahrt und Brücke

Der fünfte Bereich beinhaltet die Autobahnabfahrt 3 der A560 im Norden des Szenarios. Die L16/Bonner Straße kreuzt dort die A560. Es bestehen jeweils eine Auf- und eine Abfahrtmöglichkeit für jede Fahrtrichtung der Autobahn. In Abbildung 40 ist Kartenmaterial sowie das generierte Netzwerk zu Bereich 5 dargestellt. Folgend sind die Feststellungen des Probanden aufgelistet:

- Der Straßenverlauf des Netzwerksausschnittes entspricht dem realen Straßenverlauf.
- Der Autobahnabschnitt besteht aus drei Spuren. Da die Autobahn an dieser Stelle nur zwei Spuren besitzt stellt sich die Frage ob die dritte Spur den Standstreifen repräsentiert.
- Die Auf-/Abfahrten der L16 sind korrekt im Verkehrsnetzwerk dargestellt.
- Die Abfahrten von der A560 sind korrekt verbunden. Eine Abfahrt macht fehlerhafterweise einen Schlenker über die inneren Spuren bevor sie von der Autobahn wegführt.
- Für eine der Auffahrten, die auf die A560 führt, existiert ein Pfad welcher zur inneren Spur des Autobahnabschnittes führt. Der Pfad ist jedoch nicht mit dem Autobahnabschnitt verbunden.
- Die andere Auffahrt auf die A560 besitzt einen Pfad der auf die innere Spur des Autobahnabschnittes zuführt. Der Pfad macht jedoch kurz bevor er die innere Spur erreicht einen Schlenker und mündet in die korrekte äußere Spur.
- An der Stelle an der die L16, über eine Brücke, über die Autobahn verläuft sind korrekterweise keine Verbindungen zwischen den beiden Straßen vorhanden.

Fahrradwege

Innerhalb des definierten Szenarios existiert zwar eine Anzahl an Fahrradwegen, jedoch sind diese nicht innerhalb der generierten OpenDRIVE®-Daten vorhanden. Nach kurzer Nachforschung wurde festgestellt, dass zumindest an einigen Stellen in der OpenStreetMap Karte Fahrradwege durch einen *parallel track* Eintrag an beschriebenen Straßen beschrieben sind. Die Daten gehen jedoch an unbekannter Stelle während des Prozesses zur Erzeugung der OpenDRIVE®-Daten verloren.

Fazit des Probanden

Der Proband kam letztendlich zu dem Schluss, dass die generierten Verkehrsnetzwerke das wiedergeben was die zu Grunde liegenden Daten ermöglichen. Sämtliche Probleme und Fehler innerhalb der generierten Netzwerke sind Folge von fehlenden oder fehlerhaften Informationen in den zu Grunde liegenden Daten.

5.5.3. Auswertung gewonnener Informationen

In diesem Abschnitt sollen die durch die Befragung des Probanden gewonnenen Informationen ausgewertet werden. Dazu werden zuerst die festgestellten Fähigkeiten und anschließend sich ergebenden Probleme dargestellt. Zu den Problemen werden anschließend Fragen definiert und mögliche Antworten diskutiert.

Fähigkeiten

An dieser Stelle ist zusammengefasst, was die momentane Umsetzung leistet, um letztlich innerhalb weniger Minuten bzw. Sekunden Straßennetzwerke großer Datensätze erzeugen zu können.

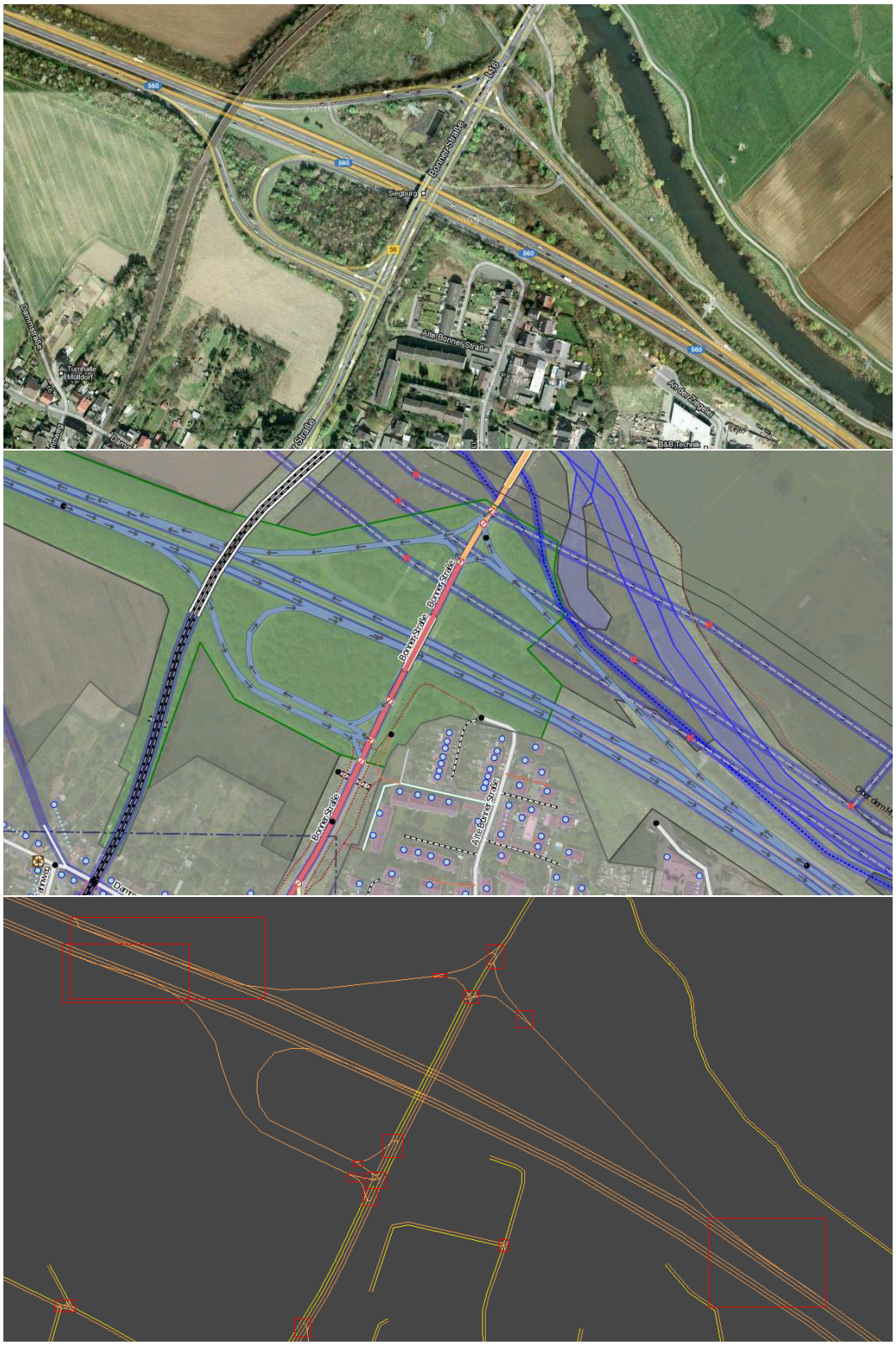


Abbildung 40: Szenario Bereich 5. Dargestellt ist eine Autobahnauf- bzw. -abfahrt im Norden des Szenarios. Oben: Autobahnabschnitt aus Google Maps. Mitte: Autobahnabschnitt aus OpenStreetMap. Unten: Verkehrsnetzwerk zum Autobahnabschnitt.

- Junction-Elemente können extrahiert und mit Informationen angereichert werden.
- Road- und Path-Elemente können extrahiert und mit Informationen angereichert werden.
- Zu jedem Road- bzw. Path-Element können mehrere Lane- und Connector-Elemente generiert und mit Informationen angereichert werden.
- Es können automatisiert Verbindungen zwischen den extrahierten Elementen gesetzt werden.
- Für die Lane- und Connector-Elemente können Waypoint-Elemente entsprechend beschriebener Geometrien generiert und innerhalb gegebener Toleranzen reduziert werden.
- Es können zusätzliche Informationen an generierte Waypoint-Elemente anfügt werden.

Probleme

- Einige kleinere Straßen oder Wege sind im generierten Netzwerk nicht vorhanden.
- Die B56 besitzt im generierten Netzwerk vier anstelle von korrekterweise zwei Spuren.
- Im generierten Netzwerk existieren offene Verbindungen an der Kreuzung in Bereich 1. Zusätzlich gibt es eine Verschiebung der anderen drei Spuren an dieser Kreuzung.
- Im generierten Netzwerk sind keine Abbiegespuren – also Fahrstreifen für abbiegende Fahrzeuge auf einer Straße – vorhanden.
- Im generierten Netzwerk existieren verbotene Abbiegepfade (z.B. auf Ausweichspur in Bereich 2).
- Im generierten Netzwerk existiert keine Unterscheidung zwischen Kreisverkehren und Kreuzungen.
- Der Autobahnabschnitt besitzt im generierten Netzwerk drei anstelle von zwei Spuren.
- Eine der Autobahnabfahrten des generierten Netzwerks macht einen Schlenker über die anderen Spuren hinweg.
- Eine der Autobahnauffahrten im generierten Netzwerk ist nicht mit dem Autobahnabschnitt verbunden.
- Die Autobahnauffahrten führen fehlerhafterweise auf die innere Spur des Autobahnabschnittes zu. Die verbundene Spur macht jedoch kurz vor Erreichen der inneren Spur einen Schlenker und mündet in die korrekte äußere Spur.
- Die Verbindungen an der Brücke in Bereich 5 mussten im Vorhinein innerhalb der Trian3D Builder Software entfernt werden.

Diskussion

1. *Warum sind einige kleinere Straßen oder Wege nicht vorhanden?*
   Dieses Problem entsteht im Verlauf des Erzeugungsprozesses für die Testdaten. Bei der manuellen Erstellung der Import Table innerhalb des Trian3D Builders müssen verschiedenste Eigenschaften (Features), welche in der Shapefile Datei beschrieben sind, einer Trian3D Builder internen Vector ID zugewiesen werden. Problematisch ist hier, dass die Identität

einzelner Features nicht deutlich ist. So sind viele unterschiedliche Namen und Typen vergeben. Durch diese Unklarheiten kann es vorkommen, dass Straßen (vor allem kleinere) gar nicht oder falsch zugewiesen und dadurch auch nicht in die OpenDRIVE®-Datei übertragen werden.

2. *Warum besitzt die B56 im St. Augustin Szenario vier anstelle von zwei Spuren?*
   Der Abschnitt in dem für die B56 vier Spuren generiert wurden ist in der OpenStreetMap Karte als *Primary road* hinterlegt. Der Abschnitt im Norden der Szenarios, wo für die B56 wiederum nur zwei Spuren generiert wurden ist als *secondary Road* hinterlegt. Alle anderen Straßen sind entweder als *Tertiary road* oder aber als *Residential road* hinterlegt. Dies legt die Vermutung nahe, dass die Anzahl der generierten Spuren von der Definition innerhalb der OpenStreetMap-Daten abhängt, obwohl dort auch die korrekte Anzahl der Fahrspuren hinterlegt ist. Dies liegt daran, wie die Trian3D Builder Software die einzelnen Straßentypen in OpenDRIVE®-Daten übersetzt. Dies geschieht anhand zugewiesener VectorIDs und dazu definierter Modifikatoren.

3. *Warum existieren die offenen Verbindungen in Bereich 1?*
   Diese und die nachfolgende Frage resultiert aus der bereits in der zweiten Frage diskutierten Problematik.

4. *Warum findet eine Verschiebung der Spuren an der Kreuzung in Bereich 1 statt?*
   Diese und die vorangehende Frage resultieren aus der bereits in der zweiten Frage diskutierten Problematik.

5. *Warum sind keine Abbiegespuren vorhanden?*
   Der erste mögliche Grund dafür, dass im generierten Netzwerk keine Abbiegespuren vorhanden sind, ist dass keine Daten dazu existieren. In OpenDRIVE® besteht die Möglichkeit entsprechende Daten abzubilden. Da dies in den vorliegenden Daten nicht der Fall ist, stellt sich die Frage, ob es in OpenStreetMap möglich ist, Abbiegespuren zu definieren. Nach kurzer Recherche wurde festgestellt, dass die Möglichkeit existiert, Abbiegespuren in OpenStreetMap zu modellieren. Auf Grund des Aufwandes bzw. um die Komplexität zu verringern, wird jedoch davon abgeraten:

   > „Da dies normalerweise ein Aufteilen des Weges erfordert, werden diese bei niederrangigen Straßen oder üblichen Kreuzungen nicht eingetragen.“ Quelle: `http://wiki.openstreetmap.org/wiki/DE:Key:lanes`

6. *Warum existieren verbotene Wege auf die Ausweichspur?*
   Stellen die verbotenen Wege überhaupt ein Problem dar?
   Die verbotenen Wege resultieren aus der Funktion zum setzen der Verbindungen innerhalb des Trian3D Builders (*Create Profile Connections*). Die Funktion teilt sich überschneidende Linien auf und fügt Kreuzungen ein wodurch die Wege entstehen. Es ist möglich diese in einem Vorverarbeitungsschritt wieder zu entfernen.

   Es soll auch angemerkt werden, dass die Wege im realen Straßennetzwerk zwar nicht existieren, theoretisch könnten die gesetzten Pfade jedoch befahren werden. Zum Beispiel dann, wenn ein Verkehrsteilnehmer an der vorangehenden Kreuzung falsch abgebogen ist und seine Fahrtroute korrigieren möchte. In dieser Situation würde also bewusst ein verbotener Weg befahren und dabei ein Risiko in Kauf genommen.

7. *Warum wird keine Unterscheidung zwischen Kreisverkehren und Kreuzungen gemacht?*
   In der OpenDRIVE®-Spezifikation ist keine Unterscheidung zwischen Kreisverkehr und Kreuzungen vorgesehen. So wird dort ein Kreisverkehr aus mehreren Straßen und Verzweigungen zusammengesetzt, was sich auch im generierten Verkehrsnetzwerk widerspiegelt.

Wie der Proband schon erkannt hat, könnte ohne eine zusätzliche Erkennung keine Unterscheidung zwischen Kreisverkehr und Kreuzung getroffen werden. Dies bietet jedoch auch Vorteile, so können diese beiden Situationen innerhalb der Logik der Agenten äquivalent verarbeitet werden und müssen nicht einzeln betrachtet werden.

8. *Warum besitzt der Autobahnabschnitt drei anstelle von zwei Spuren?*
   Die erste Vermutung war, dass die zusätzliche dritte Spur den Standstreifen entlang der Fahrbahnen (auf welchem sich auch die Beschleunigungs- und Abfahrtstreifen befinden) repräsentiert. Wirft man jedoch einen Blick in die OpenDRIVE®-Datei stellt man fest, dass dort sogar fünf Spuren definiert sind. Zwei der fünf Spuren –eine links und eine rechts der anderen drei Spuren– sind explizit mit *shoulder* also Standstreifen gekennzeichnet. Die zweite Vermutung ist, dass die zugewiesene VectorID innerhalb des Trian3D Builders für Autobahnen automatisch drei Spuren erzeugt.

9. *Warum macht eine der Autobahnabfahrten einen Schlenker über die anderen Spuren?*
   Das Problem an dieser Stelle lässt sich darauf zurückführen, dass die Width-Einträge innerhalb der OpenDRIVE®-Datei, welche für den Abstand der Spuren zur Referenzline verantwortlich sind, falsche Daten enthalten. Um dies zu beheben müsste an der entsprechenden Kreuzung innerhalb des Trian3D Builders eine andere VectorID zugewiesen werden deren Modifikatoren zu korrekten Daten führen.

10. *Warum ist eine der Autobahnauffahrten nicht mit dem Autobahnabschnitt verbunden?*
    Die fehlende Verbindung scheint auf einen Fehler in den OpenStreetMap Daten zurückgeführt werden zu können. So ist die Autobahn an der Stelle der fehlenden Verbindung nicht aufgeteilt, wie sie es bei der anderen Autobahnauffahrt ist.

11. *Warum laufen die Autobahnauffahrten jeweils auf die innere Spur der Autobahn zu und warum macht die verbundene Spur dann doch einen Schlenker und mündet in die korrekte äußere Spur?*
    Dieses Problem scheint ebenfalls aus falschen Daten in der OpenDRIVE®-Datei zu resultieren. Es sieht so aus als wenn die Referenzlinie der Auffahrt sich der Referenzlinie des Autobahnabschnittes annähert und kurz bevor sie einander erreichen einen Schlenker zur richtigen Spur macht. Dies deutet auf fehlerhafte Daten in den Geometry-Einträgen sowie den Witdth-Einträgen hin.

12. *Warum befinden sich an der Brücke in Bereich 5 Verbindungen wenn diese nicht vorher in der Tiran 3D Builder Software entfernt wurden?*
    Wie in einer der vorherigen Fragen bereits angemerkt wurde, setzt die Trian3D Builder interne Funktion Verbindungen an sämtlichen Schnittpunkten. Da keine Informationen über die Höhen der Straßen vorliegen, kann die Stelle an der die Brücke über die Autobahn hinwegführt nicht automatisch von einer normalen Kreuzung unterschieden werden.

Fazit

Nach dem das Evaluierungsszenario und das dazu generierte Verkehrsnetzwerk ausführlich betrachtet und die bestehenden Probleme analysiert wurden können mehrere grundsätzliche Problemstellungen festgehalten werden.

- Die Qualität der zur Generierung der Verkehrsnetzwerke genutzten Daten lässt es momentan noch nicht zu, Netzwerke zu generieren, die sofort bzw. ohne weitere manuelle Bearbeitung genutzt werden können.

- Die Probleme liegen nicht an der OpenDRIVE®-Spezifikation. Diese lässt Beschreibungen mit hinreichendem Detailgrad zur Generierung von Verkehrsnetzwerken für die Ziel-

Umgebung zu.

- Der Überführungsprozess der OpenStreetMap Daten hin zu OpenDRIVE®-Daten durch die Trian3D Builder Software ist nicht hinreichend bekannt. So bestehen Probleme und Unklarheiten mit der manuellen Zuweisung von Features, der heuristischen Erzeugung von Verbindungen und der Interpretation von einzelnen Straßentypen.

- Die OpenStreetMap Daten sind nicht ausführlich genug. So bilden sie z.B. in der Regel keine Abbiegespuren ab.

Weiterhin gilt es mehrere Fragestellungen zu untersuchen. Dabei ist festzustellen wo genau Informationen verloren gehen oder ob diese erst gar nicht existieren. Zusätzlich kann untersucht werden wo im Workflow zur Datengenerierung Optimierungsbedarf besteht. Ziel ist es mehr Informationen verfügbar machen zu können. Letztendlich stellt sich die Frage, ob vorteilhaftere Verfahren z.B. mit weniger Überführungsschritten existieren, mit denen es möglich ist, detailliertere OpenDRIVE®-Daten zu gewinnen. Es sei angemerkt, dass dies keine Modelländerungen nach sich ziehen würde, da die Modelle und die Generierungsprozesse für die Netzwerke davon unabhängig sind.

6. Zusammenfassung

Im Rahmen dieser Arbeit mussten mehrere Probleme bearbeitet werden, um bestehende, im OpenDRIVE®-Format abgespeicherte Daten in der genutzten Simulationsumgebung nutzbar zu machen. Nachdem im Vorfeld dieser Arbeit bereits eine Schnittstelle realisiert wurde, die es ermöglicht xml-basierte OpenDRIVE®-Daten in die Simulationsumgebung zu importieren, musste eine Klassenstruktur geschaffen werden, mit der die Daten nutzbar gemacht werden konnten. Anschließend galt es sich in die OpenDRIVE®-Spezifikation einzuarbeiten und die bestehenden Strukturen und die daraus resultierenden Möglichkeiten zu analysieren. Darauf aufbauend mussten Methoden implementiert werden, um die Daten nutzbar zu machen. Der anspruchsvollste Teil dabei war die Umsetzung der geometrischen Berechnungen für die beschriebenen Straßenreferenzlinien.

Nachdem die Daten nutzbar gemacht wurden, sollten diese in Form eines Verkehrsnetzwerks hinterlegt werden, welches von Agenten zur Navigation und zur Informationsgewinnung genutzt werden kann. Dazu wurden bestehende Verkehrsnetzwerke aus anderen virtuellen Umgebungen, wie z.B. aus Projekten zu virtueller Realität oder Videospielen, betrachtet und deren allgemeiner Aufbau analysiert. Es stellte sich jedoch heraus, dass Informationen zu bestehenden Verkehrsnetzwerkmodellen relativ rar sind und für die wenigen, die in der Literatur zu finden sind, nur relativ spärliche Informationen zur Verfügung stehen. Anschließend musste eine Einarbeitung in im Projekt bereits bestehende Abläufe erfolgen, so bestand noch kein konkretes Verkehrsnetzwerkmodell, jedoch einzelne realisierte Elemente, die übernommen werden sollten.

Schließlich wurde ein eigenes Netzwerkmodell definiert. Dieses nutzt die Vorteile bestehender Verkehrsnetzwerkmodelle und ist gleichzeitig so geschaffen, dass bestehende Abläufe weiterhin genutzt werden können und nicht vollständig neu realisiert werden mussten. Zur Definition des Gesamtmodells mussten einzelne Netzwerkelemente definiert werden, wozu elementare Fragen geklärt werden mussten: Welche Daten muss ein Verkehrsnetzwerk für die Agenten zu Verfügung stellen? In welcher Weise navigieren die Agenten auf dem Netzwerk? An welcher Stelle innerhalb der Netzwerke ist es möglich, zusätzliche Informationen zu hinterlegen?

Nach der Definition des Verkehrsnetzwerkmodells, wurden die einzelnen Elemente des Modells implementiert. Darüber hinaus wurde ein Überführungsprozess implementiert, der aus Daten im OpenDRIVE®-Format Verkehrsnetzwerke des definierten Modells erzeugt. Dabei bestand das Problem, dass bestehende Testdaten (nach [12]) nicht alle Funktionalitäten der OpenDRIVE®-Spezifikation ausschöpfen und so nur genutzte Attribute berücksichtigt werden konnten. Des weiteren wäre eine Berücksichtigung der vollständigen Spezifikation wenig sinnvoll gewesen, da nicht alle Aspekte relevant für das Projekt sind. Zusätzlich hätte der Zeitrahmen nicht ausgereicht, um sämtliche Möglichkeiten zu berücksichtigen. Folglich wurde nur eine Teilmenge der für das Projekt wichtigen Attribute zur Erzeugung der Verkehrsnetzwerke berücksichtigt.

Der letzte Schritt des Projektes bestand in der Evaluierung der umgesetzten Funktionalitäten. Diese fand in mehreren Schritten statt. Zuerst wurden Eigenschaften über die während der Entwicklung genutzten Datensätze und über das zur Evaluation genutzte Computersystem festgehalten. Anschließend wurden Leistungsmerkmale zum Generierungsprozess der Verkehrsnetzwerke ermittelt und eine Überprüfung der geometrischen Berechnungen vollzogen. Danach folgte die Evaluation der erzeugten Netzwerke, welche aus zwei Schritten bestand. Der erste Schritt bestand darin bestehende Agenten auf den generierten Verkehrsnetzwerken agieren zu lassen und somit die direkte Nutzbarkeit der Netzwerke und des unterliegenden Modells zu bestätigen. Im zweiten Schritt wurden Testdaten zu einem neu definierten Szenario, welches eine Anzahl geforderter Eigenschaften erfüllte, erzeugt und ein zugehöriges Netzwerk generiert. Um die Qualität des implementierten Generierungsprozesses zu evaluieren, wurde dieses Netzwerk anschließend durch einen Probanden mit realen Straßendaten verglichen und Probleme aufgezeigt.

Bei der Evaluation des Netzwerks der mesoskopsichen Simulationsebene konnte gezeigt werden,

dass es möglich ist, Simulationen auf generierten Netzwerken der mesoskopischen Ebene auszuführen. Dabei wurde festgestellt, dass keine Deadlocks auftreten und dass die Netzwerkselemente gleichmäßig durch die Agenten genutzt werden. Gleichzeitig konnte gezeigt werden, dass eine plausible Durchschnittswartezeit entsteht. Bei der Evaluation des Netzwerks der mikroskopischen Simulationsebene konnten erfolgreich Simulationen mit Agenten ausgeführt werden, ohne dass Fehler auftraten. Wie bei der mesoskopischen Simulation konnte auch hier eine ausgewogene Nutzung der Netzwerkelemente erzielt werden. Für einige Agenten konnte gezeigt werden, dass es möglich ist, längere Zeit im Netzwerk zu agieren, ohne ein offenes Ende zu erreichen, was eine gewisse Abgeschlossenheit des Netzwerks bestätigt. Bei der Evaluation der Qualität der Netzwerke wurde festgestellt, dass die Qualität der Netzwerke zu niedrig ist, um diese ohne manuelle Anpassungen in einer fertigen Anwendung nutzen zu können. Dies konnte jedoch in allen Fällen auf die zu Grunde liegenden Daten zurückgeführt werden.

6.1. Ausblick

Trotz der positiven Ergebnisse dieses Projektes, existieren weiterhin Probleme in die in Zukunft Arbeit investiert werden kann. Der in Kapitel 4.4.2 angedachte Schritt 4 zur Erzeugung von zusätzlichen Waypoint-Elementen mit zusätzlichen Informationen sollte umgesetzt werden. Zusätzlich sollten mehrere in der momentanen Umsetzung noch auf die Lane-Element bezogene Informationen (z.B. linkes und rechtes Nachbarelement) in die Waypoint-Elemente verlagert werden. Zusätzlich bestehen einige Ideen für weitere Forschungsprojekte, welche in den folgenden Abschnitten beschrieben werden.

6.1.1. Weiterentwicklung des Rapid Scenario Generation Workflow

Die erste Idee ist es, weitere Entwicklungszeit in den in [12] eingeführten Arbeitsablauf zur schnellen Szenario Generierung zu investieren. Dabei können mehrere Fragen gestellt werden:

- Wie können Fehler in den zu Grunde liegenden Daten – seien es die OpenStreetMap Daten oder die OpenDRIVE®-Daten – detektiert und behoben werden?
- Wie ist es möglich im Vorhinein weitere nicht vorhandene oder fehlende Informationen zu ergänzen?
- Wie sollte ein Interface aussehen, mit dem manuell Erweiterungen und Optimierungen an den generierten Verkehrsnetzwerken vorgenommen werden können?

Der momentane Arbeitsablauf ist in Abbildung 41 noch einmal dargestellt.

6.1.2. Zusammenführung der Verkehrsnetzwerkmodelle beider Simulationsebenen

Zusätzlich besteht die Idee die Verkehrsnetzwerkmodelle für die beiden Simulationsebenen (mikroskopisch und mesoskopisch) zusammenzuführen bzw. die abstraktere mesoskopische Simulation mit Daten aus dem Verkehrsnetzwerk der mikroskopischen Simulationsebene zu betreiben. Dies würde einige Vorteile mit sich bringen. Innerhalb der Simulationsumgebung bestünden weniger Elemente, welche erstens Speicherplatz belegen und zweitens verwaltet werden müssten. Durch die stärkere Kopplung zwischen den Netzwerkmodellen können Informationen gemeinsam von den Agenten beider Simulationsansätze genutzt werden und müssen nicht mehrfach hinterlegt werden. Zusätzlich müsste statt zweier Verkehrsnetzwerke nur noch eines gepflegt werden. Hierbei ist auch wichtig, dass durch Änderungen an einem Netzwerk keine Inkonsistenzen mehr gegenüber dem anderen Netzwerk entstehen können. Ein weiterer entscheidender Vorteil wäre, dass der Transferprozess von Agenten zwischen den beiden Ebenen wesentlich vereinfacht wird.

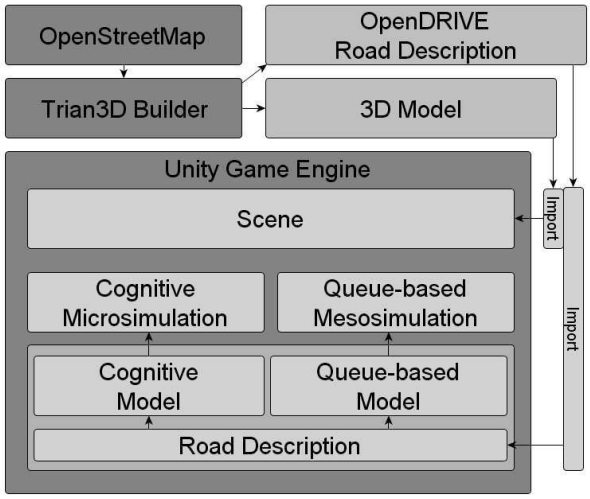


Abbildung 41: Arbeitsablauf zur Realisierung einer schnellen Szenario Generierung. OpenStreetMap Daten werden von der Trian3D Builder Software verwendet um eine Straßenbeschreibungsdatei im OpenDRIVE®-Dateiformat zu erzeugen. Diese Datei wird in die Simulationsumgebung (Unity) importiert, wo sie genutzt wird, um die verschiedenen Verkehrsnetzwerke zu generieren. Bildquelle: [12].

Um eine solche Vereinigung zu realisieren ist es möglich das Modell für die mesoskopische Simulationsebene in das Modell für die mikroskopische Simulationsebene zu integrieren. Dabei müssten die Eigenschaften der Node-Elemente in die Junction-Elemente und die Eigenschaften der Edge-Elemente in die Road-Elemente integriert werden.

6.1.3. Weiterentwicklung der kognitiven Agenten

Weiterhin kann Arbeit in die Weiterentwicklung der Agenten der mikroskopischen Simulationsebene investiert werden. Die kognitiven Eigenschaften dieser können weiterentwickelt werden. Das Hauptziel bei der Weiterentwicklung der kognitiven Agenten ist es den Realismus der Simulation zu erhöhen. Dabei sollten die Ziele des AVeSi Projektes (siehe 2.5) beachtet werden. In den folgenden Abschnitten werden zwei Ideen vorgestellt, die zu einer Verbesserung dieser Ziele beitragen könnten.

Aktivitäten- und Routenplanung

Über die in dieser Arbeit eingeführte Art von Waypoint-Elementen ist es möglich Informationen an sämtlichen Stellen im Netzwerk zu hinterlegen. Dadurch ist es möglich Orte mit potentiellem Interesse für die Agenten zu vermerken. Dies könnten z.B. Orte wie Parkplätze, Einkaufszentren oder Wohngebiete sein. Es kann nun eine Liste mit diesen, innerhalb eines Szenarios vermerkten Orten erstellt werden. Durch die den Orten zugewiesenen Waypoint-Elemente besitzt jeder Ort zusätzlich eine Position im Straßennetzwerk. Darauf aufbauend könnte eine Zielbestimmung für einzelne im Netzwerk agierende Agenten ausgeführt werden. In Folge dessen kann für diese Ziele eine Routenplanung durchgeführt werden, mit welcher es letztendlich möglich ist die Agenten gezielt zu ausgewählten Orten im Netzwerk navigieren zu lassen. Durch ein solches Vorgehen sollte es möglich sein den Realismus der Bewegungen von Agenten innerhalb des Netzwerks zu steigern und somit nachvollziehbare Verkehrsflüsse entstehen zu lassen.

Literaturverzeichnis

[1] Christopher S. Applegate, Stephen D. Laycock, and Andy M. Day. Real-Time Traffic Simulation Using Cellular Automata. *EG UK Theory and Practice of Computer Graphics*, 11:91–98, 2010.

[2] A. Aw and Michel Rascle. Resurrection of "Second Order" Models of Traffic Flow. *SIAM journal on applied mathematics*, 60(3):916–938, 2000.

[3] Gari Biasillo. Representing a Racetrack for the AI. *AI Game Programming Wisdom. Charles River Media*, pages 439–443, 2002.

[4] H. J. Bungartz, S. Zimmer, M. Buchholz, and D. Pflüger. *Modellbildung und Simulation: Eine anwendungsorientierte Einführung*. Springer, 2009.

[5] Thomas Dettmar. Queuing Models for Traffic Simulations in Virtual Environments. Master's thesis, Hochschule Bonn-Rhein-Sieg, 2013.

[6] D. Douglas and T. Peucker. Algorighms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. In *Cartographica: International Journal for Geographic Information and Geovisualization 10.2*, pages 112–122. 1973.

[7] Marius Dupuis et al. OpenDRIVE® Format Specification. `http://www.opendrive.org/docs/OpenDRIVEFormatSpecRev1.3D.pdf`, 2011.

[8] Christian Gawron. *Simulation-Based Traffic Assignment*. PhD thesis, University of Cologne, Germany, 1998.

[9] A. P. Gerdelan. A Solution for Streamlining Intelligent Agent-Based Traffic into 3D Simulations and Games. Technical Report CSTN-072, Massey University, Albany, North Shore 102-904, Auckland, New Zealand, January 2009.

[10] A. P. Gerdelan. Driving Intelligence: A New Architecture and Novel Hybrid Algorithm for Next-Generation Urban Traffic Simulation. Technical Report CSTN-079, Massey University, Albany, North Shore 102-904, Auckland, New Zealand, February 2009.

[11] John Hamill and Carol O'Sullivan. Virtual dublin: a framework for real-time urban simulation. *Journal of WSCG*, 11, 2003.

[12] Tobias Haubrich, Sven Seele, Rainer Herpers, Martin E. Müller, and Peter Becker. Semantic Road Network Models for Rapid 3D Traffic Scenario Generation. In *Tagungsband ASIM/GI-Fachgruppentreffen STS/GMMS, Workshop Simulation technischer Systeme - Grundlagen und Methoden in Modellbildung und Simulation*, pages 51–55. Arbeitsgemeinschaft Simulation ASIM, 2013.

[13] Dirk Helbing and Martin Treiber. Gas-Kinetic-Based Traffic Model Explaining Observed Hysteretic Phase Transition. *Phys. Rev. Lett.*, 81:3042–3045, Oct 1998.

[14] Michael Hinterseher. Entwicklung von Konzepten, Algorithmen und Optimierungsverfahren zur Transformation von Knoten in einem Netzwerk unter Beachtung von Integritätsbedingungen. Diplomarbeit, Fachhochschule München, 1999.

[15] Jan Kratochvíl. Living City in Mafia II. Game Developers Conference Europe 2010, 2010.

[16] Jan Kratochvíl. Creating Living City for Openworld Game. Vienna Game/AI Conference 2012, 2012.

[17] Michael J. Lighthill and Gerald Beresford Whitham. On Kinematic Waves. II. A theory of Traffic Flow on Long Crowded Roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 229(1178):317–345, 1955.

[18] Kai Nagel and Michael Schreckenberg. A Cellular Automaton Model for Freeway Traffic. *Journal de Physique I*, 2(12):2221–2229, 1992.

[19] Bryan Raney, Andreas Voellmy, Nurhan Cetin, Milenko Vrtic, and Kai Nagel. Towards a Microscopic Traffic Simulation of All of Switzerland. In PeterM.A. Sloot, AlfonsG. Hoekstra, C.J.Kenneth Tan, and JackJ. Dongarra, editors, *Computational Science - ICCS 2002*, volume 2329 of *Lecture Notes in Computer Science*, pages 371–380. Springer Berlin Heidelberg, 2002.

[20] Paul I. Richards. Shock Waves on the Highway. *Operations research*, 4(1):42–51, 1956.

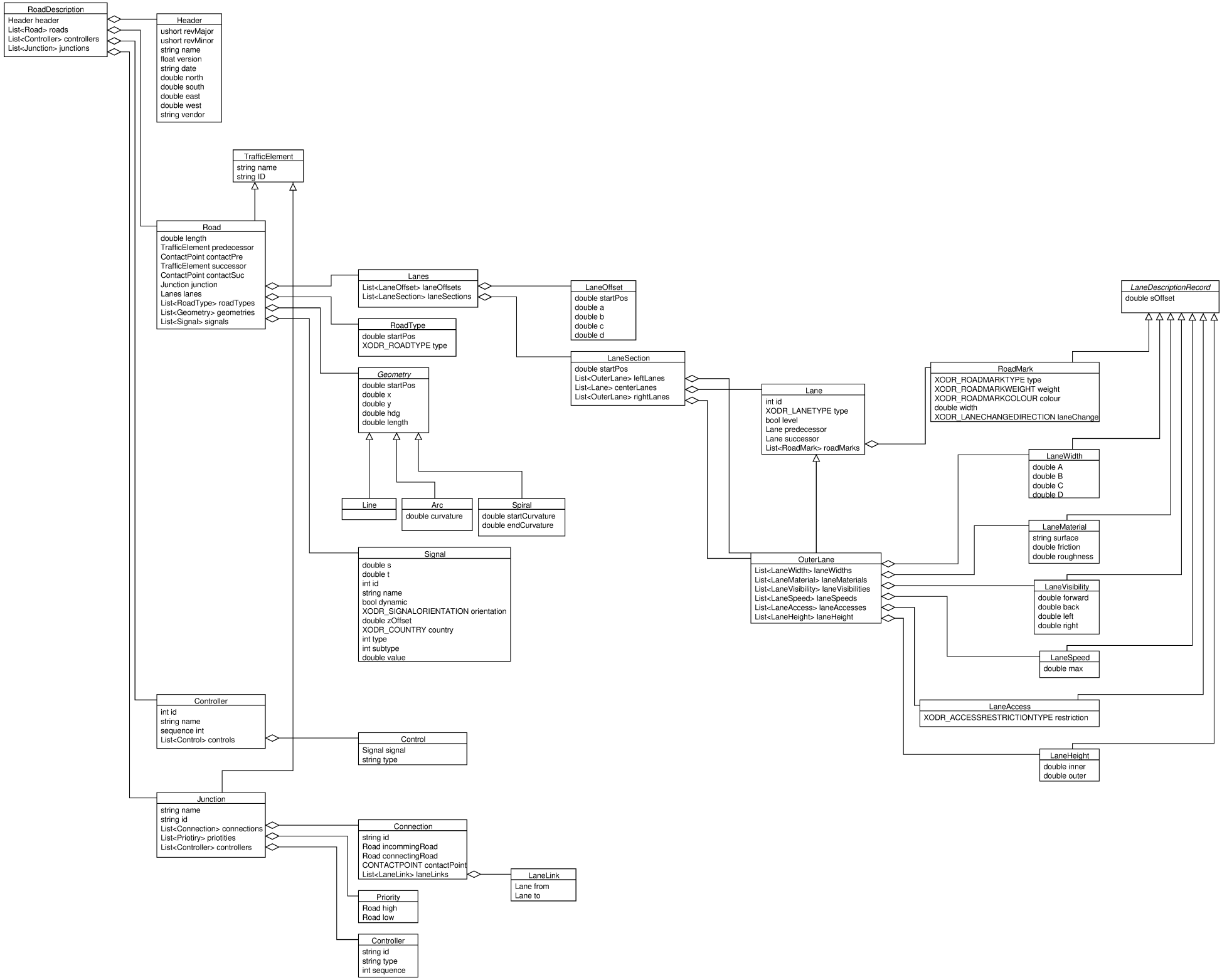
[21] Martin Treiber and Dirk Helbing. Realistische Mikrosimulation von Strassenverkehr mit einem einfachen Modell. In *16th Symposium Simulationstechnik ASIM*, volume 2002, page 80, 2002.

[22] Unity Technologies. Unity Script Reference. `http://docs.unity3d.com/Documentation/ScriptReference/`.

[23] VIRES Simulationstechnologie GmbH. OpenDRIVE®-Webseite. `http://www.opendrive.org/`, 2011.

A. Anhang

- S. 71
  UML-ähnliche Darstellung der objektorientierten Datenstruktur für die aus OpenDRIVE®-Dateien eingelesenen Daten.


RoadDescription
Header header
List<Road> roads
List<Controller> controllers
List<Junction> junctions
Header
ushort revMajor
ushort revMinor
string name
float version
string date
double north
double south
double east
double west
string vendor
TrafficElement
string name
string ID
Road
double length
TrafficElement predecessor
ContactPoint contactPre
TrafficElement successor
ContactPoint contactSuc
Junction junction
Lanes lanes
List<RoadType> roadTypes
List<Geometry> geometries
List<Signal> signals
Lanes
List<LaneOffset> laneOffsets
List<LaneSection> laneSections
LaneOffset
double startPos
double a
double b
double c
double d
RoadType
double startPos
XODR_ROADTYPE type
Geometry
double startPos
double x
double y
double hdg
double length
Line
Arc
double curvature
Spiral
double startCurvature
double endCurvature
LaneSection
double startPos
List<OuterLane> leftLanes
List<Lane> centerLanes
List<OuterLane> rightLanes
Lane
int id
XODR_LANETYPE type
bool level
Lane predecessor
Lane successor
List<RoadMark> roadMarks
LaneDescriptionRecord
double sOffset
RoadMark
XODR_ROADMARKTYPE type
XODR_ROADMARKWEIGHT weight
XODR_ROADMARKCOLOUR colour
double width
XODR_LANECHANGEDIRECTION laneChange
LaneWidth
double A
double B
double C
double D
LaneMaterial
string surface
double friction
double roughness
LaneVisibility
double forward
double back
double left
double right
LaneSpeed
double max
LaneAccess
XODR_ACCESSRESTRICTIONTYPE restriction
LaneHeight
double inner
double outer
OuterLane
List<LaneWidth> laneWidths
List<LaneMaterial> laneMaterials
List<LaneVisibility> laneVisibilities
List<LaneSpeed> laneSpeeds
List<LaneAccess> laneAccesses
List<LaneHeight> laneHeight
Signal
double s
double t
int id
string name
bool dynamic
XODR_SIGNALORIENTATION orientation
double zOffset
XODR_COUNTRY country
int type
int subtype
double value
Controller
int id
string name
sequence int
List<Control> controls
Control
Signal signal
string type
Junction
string name
string id
List<Connection> connections
List<Priotiry> priotities
List<Controller> controllers
Connection
string id
Road incommingRoad
Road connectingRoad
CONTACTPOINT contactPoint
List<LaneLink> laneLinks
LaneLink
Lane from
Lane to
Priority
Road high
Road low
Controller
string id
string type
int sequence